Hidden Markov Model Cheat Sheet

Barak A. Pearlmutter

(GIT ID: `badf11790325fa3e96a6c58ef42808a0fe99a530`)

This document is a "cheat sheet" on Hidden Markov Models (HMMs). It resembles lecture notes, except that it cuts to the chase a little faster by defining terms and divulging the useful formulas as quickly as possible, in the place of gentle explanations and intuitions.

1 Notation

HMM:

- states are not observable.
- observations are probabilistic function of state
- state transitions are probabilistic

N**:** number of hidden states, numbered $1, \ldots, N$
M**:** number of output symbols, numbered $1, \ldots, M$
T**:** number of time steps in sequence of states and sequence of output symbols
$\vec{q}$**:** sequence of states traversed, $\vec{q} = (q_1, \ldots, q_t, \ldots, q_T)$ where each $q_t \in \{1, \ldots, N\}$
$\vec{o}$**:** observed output symbol sequence, $\vec{o} = (o_1, \ldots, o_t, \ldots, o_T)$ where $o_t \in \{1, \ldots, M\}$
A: state transition matrix, $a_{ij} = P(q_{t+1} = j \,|\, q_t = i)$
B**:** per-state observation distributions, $b_i(k) = P(o_t = k \,|\, q_t = i)$
$\vec{\pi}$**:** initial state distribution, $\pi_i = P(q_1 = i)$
λ**:** all numeric parameters defining the HMM considered together, $\lambda = (\mathbf{A}, B, \vec{\pi})$
indices: i, j index states; k indexes output symbols; t indexes time

We proceed to review the solutions to the three big HMM problems: finding $P(\vec{o} \,|\, \lambda)$, finding $\vec{q}^* = \operatorname{argmax}_{\vec{q}} P(\vec{q} \,|\, \vec{o}, \lambda)$, and finding $\lambda^* = \operatorname{argmax}_{\lambda} P(\vec{o} \,|\, \lambda)$.

2 Probability of sequence of observations

We wish to calculate $P(\vec{o} \,|\, \lambda)$.

Definition: $\alpha_t(i) = P(o_1, \ldots, o_t, q_t = i \mid \lambda)$. (In words: the probability of observing the head of length t of the observations and being in state i after that.)

Initialization: $\alpha_1(i) = \pi_i \, b_i(o_1)$.

Loop: $\alpha_{t+1}(j) = \left(\sum_{i=1}^{N} \alpha_t(i) \, a_{ij} \right) b_j(o_{t+1})$

At termination, $P(\vec{o} \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$.

Note: complexity is $\mathcal{O}(N^2 T)$ time, $\mathcal{O}(NT)$ space.

Note: calculating the α values is called the "forward algorithm."

3 Optimal state sequence from observations

Find $\vec{q}^* = \operatorname{argmax}_{\vec{q}} P(\vec{q} \mid \vec{o}, \lambda)$, the most likely sequence of hidden states given the observations.

Note: calculating the most likely sequence of states is called a "Viterbi alignment."

Definition: $\beta_t(i) = P(o_{t+1}, o_{t+2}, \ldots, o_T \mid q_t = i, \lambda)$. (In words: the probability that starting in state i at time t, then generating the remaining tail of the observations.)

Initialization: $\beta_T(i) = 1$.

Loop: $\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)$. Calculated backwards: $t = T-1, T-2, \ldots, 1$.

Note: calculating the β values is called the "backward algorithm."

Define:

$$\delta_t(i) = \max_{q_1, \cdots, q_{t-1}} P(q_1, \ldots, q_{t-1}, q_t = i, o_1, \ldots, o_t \mid \lambda).$$

(In words: the probability of generating the head of length t of observables and having gone through the most likely states for the first $t-1$ steps and ending up in state i.)

Initialization: $\delta_1(i) = \pi_i \, b_i(o_1)$

Loop: $\delta_t(j) = (\max_i \delta_{t-1}(i) \, a_{ij}) \, b_j(o_t)$

Initialization: $\psi_1(i) = 0$

Loop: $\psi_t(j) = \operatorname{argmax}_i \delta_{t-1}(i) \, a_{ij}$

Termination: $P^* = \max_i \delta_T(i)$, the probability of generating the entire sequence of observables via the most probable sequence of states.

Termination: $q_T^* = \operatorname*{argmax}_i \delta_T(i)$, the most probable final state.

Loop to find state sequence ("backtracking"): $q_t^* = \psi_{t+1}(q_{t+1}^*)$

Note: ψ is written "psi" in English, and pronounced "p'sai."

3.1 Useful property of α and β

Note that

$$\begin{aligned}\sum_i \alpha_t(i)\,\beta_t(i) &= \sum_i P(o_1,\ldots,o_t,q_t=i\,|\,\lambda)\,P(o_{t+1},o_{t+2},\ldots,o_T\,|\,q_t=i,\lambda)\\ &= \sum_i P(o_1,\ldots,o_t,o_{t+1},o_{t+2},\ldots,o_T,q_t=i\,|\,\lambda)\\ &= \sum_i P(\vec{o},q_t=i\,|\,\lambda)\\ &= P(\vec{o}\,|\,\lambda)\end{aligned}$$

This logic holds for any t, so the given sum should be the same for any t. (The earlier formula for $P(\vec{o}\,|\,\lambda)$ was for the special case $t = T$ since $\beta_T(i) = 1$.) This formula thus provides a useful debugging test for HMM programs.

4 Estimate model parameters

Given $\vec{o}$ find $\lambda^* = \operatorname{argmax}_\lambda P(\vec{o}\,|\,\lambda)$.

Not an analytic solution. Instead, we start with a guess of λ, typically random, then iterate λ to a local maximum, using an EM algorithm. At each step we "re-estimate" a new λ, called $\hat{\lambda}$, which has an increased probability of generating $\vec{o}$. (Or if already at a (possibly local) optimum, the same probability.)

Note: this process is called "Baum-Welch Re-Estimation."

Typical stopping rule for this re-estimation loop is:

$$\text{stop when} \quad \log P(\vec{o}\,|\,\hat{\lambda}) - \log P(\vec{o}\,|\,\lambda) < \epsilon \quad \text{for some small } \epsilon$$

Note: debugging hint, $P(\vec{o}\,|\,\hat{\lambda}) \geq P(\vec{o}\,|\,\lambda)$ should always be true.

Definition: $\gamma_t(i) = P(q_t = i\,|\,\vec{o},\lambda)$. (In words: the probability of having been in state i at time t.)

$$\gamma_t(i) = \frac{\alpha_t(i)\,\beta_t(i)}{P(\vec{o}\,|\,\lambda)}$$

Definition: $\xi_t(i,j) = P(q_t = i, q_{t+1} = j \mid \vec{o}, \lambda)$. (In words: the probability of having transitioned from state i to j at time t.)

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij}\, b_j(o_{t+1})\, \beta_{t+1}(j)}{P(\vec{o} \mid \lambda)}$$

Note: $\sum_i \gamma_t(i) = 1$ and $\sum_i \sum_j \xi_t(i,j) = 1$.

Note: ξ is written "xi" in English, and pronounced "k'sai."

We write "#" to abbreviate the phrase "expected number of times"

# state i visited: $\displaystyle\sum_{t=1}^{T} \gamma_t(i)$

# transitions from state i to state j is: $\displaystyle\sum_{t=1}^{T-1} \xi_t(i,j)$

$$\hat{\pi}_i = \frac{\gamma_1(i)}{\displaystyle\sum_j \gamma_1(j)} = \gamma_1(i)$$

$$\hat{a}_{ij} = \frac{\text{\# transitions state } i \text{ to state } j}{\text{\# transitions from state } i} = \frac{\displaystyle\sum_{t=1}^{T-1} \xi_t(i,j)}{\displaystyle\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$\hat{b}_j(k) = \frac{\text{\# in state } j \text{ and output symbol } k}{\text{\# in state } j} = \frac{\displaystyle\sum_{t=1}^{T} [o_t = k]\, \gamma_t(j)}{\displaystyle\sum_{t=1}^{T} \gamma_t(j)}$$

where we use Knuth notation, $[\textit{boolean\_condition}] = 1$ or 0 depending on whether *boolean_condition* is true or false.

4.1 Training on multiple sequences

The above is for *one* output observable sequence $\vec{o}$. If there are multiple such observable output sequences, *i.e.* a training set of them, then the basic variables defined above (α, β, etc) are computed for each of them. Except for the re-estimation formulas, which need to sum over them as an "outer" sum around the sums shown.

We use a superscript (p) to indicate values computed for observable sequence $\vec{o}^{(p)}$. Note that λ and N and M are independent of p, but T is not since each string in the training set might be a different length, $T^{(p)} = \dim \vec{o}^{(p)}$.

The update formulas become:

$$\hat{\pi}_i = \frac{\sum_p \gamma_1^{(p)}(i)}{\sum_p 1}$$

$$\hat{a}_{ij} = \frac{\text{\# transitions state } i \text{ to state } j}{\text{\# transitions from state } i} = \frac{\sum_p \sum_{t=1}^{T^{(p)}-1} \xi_t^{(p)}(i,j)}{\sum_p \sum_{t=1}^{T^{(p)}-1} \gamma_t^{(p)}(i)}$$

$$\hat{b}_j(k) = \frac{\text{\# in state } j \text{ and output symbol } k}{\text{\# in state } j} = \frac{\sum_p \sum_{t=1}^{T^{(p)}} [o_t^{(p)} = k]\, \gamma_t^{(p)}(j)}{\sum_p \sum_{t=1}^{T^{(p)}} \gamma_t^{(p)}(j)}$$