Course: Model, Learning, and Inference: Lecture 6

Alan Yuille
Department of Statistics, UCLA
Los Angeles, CA 90095
yuille@stat.ucla.edu

Abstract

Hidden Markov Models (to illustrate DP and EM)
NOTE: NOT FOR DISTRIBUTION!!

1 Introduction

Previous lecture showed the general method for learning distributions with hidden variables. In this lecture we make this precise by describing Hidden Markov Models (HMMs). This will involve dynamic programming and EM.

HMMs were developed for speech processing in the 1970's but have been used for an enormous range of other applications including vision, see handout.

2 Observable Markov Models

Observable Markov models have the following ingredients. A set of N distinct *hidden states* $\{s_1, ..., s_N\}$. Denote the state at time t by q_t, where $q_t = s_i$ means the system is in state s_i at time t.

There is a distribution on the sequence of states. This can be formulated generally as $P(q_{t+1} = s_j | q_t = s_i, q_{t-1} = s_k, ...)$. In this lecture we assume a first-order Markov model so that:

$$P(q_{t+1} = s_j | q_t = s_i, q_{t-1} = s_k, ...) = P(q_{t+1} = s_j | q_t = s_i). \tag{1}$$

I.e., the future is independent of the past except for the proceeding time state.

A first-order Markov model is specified by the *transition probabilities* $a_{ij} = P(q_{t+1} = s_j | q_t = s_i)$ which obey $a_{ij} \geq 0, \ \forall i, j$ and $\sum_j a_{ij} = 1, \ \forall j$. Denote these transition probabilities by A. The model also requires *initial probabilities* $\pi_i = P(q_1 = s_i)$ with $\sum_{i=1}^N \pi_i = 1$. Denoted by π.

For an observable Markov model we can directly observe the states $\{q_t\}$. An observation sequence $O = Q = \{q_1, ..., q_T\}$. We can directly compute the probability of this sequence to be:

$$P(O = Q | A, \pi) = P(q_1) \prod_{t=2}^{T} P(q_t | q_{t-1}) = \pi_{q_1} a_{q_1 q_2} ... a_{q_{T-1} q_T}. \tag{2}$$

We can learn the transition and initial probabilities by maximum likelihood (ML) from a set of observation sequences $\{O^k : k = 1, ..., K\}$:

$$(A^*, \pi^*) = \arg\max_{(A,\pi)} \prod_{k=1}^{K} P(O^k | A, \pi). \tag{3}$$

This can be directed computed to give:

$$\pi_i^* = \frac{\sum_{k=1}^K I(q_1^k = s_i \text{ AND } q_{t+1}^k = s_j)}{K},$$
$$a_{ij}^* = \frac{\sum_{k=1}^K I(q_t^k = s_i \text{ AND } q_{t+1}^k = s_j)}{\sum_{k=1}^K \sum_{t=1}^{T-1} I(q_t^k = s_i)}. \tag{4}$$

Here $I(q = s)$ is the indicator function – $I(q = s) = 1$ if $q = s$, $I(q = s) = 0$ if $q \neq s$.

How is this related to exponential models of form $P(\vec{x}|\vec{\lambda}) = (1/Z[\vec{\lambda}]) \exp\{\vec{\lambda} \cdot \vec{\phi}(\vec{x})\}$ and standard ML learning? I.e. solve $\vec{\lambda}^* = \arg\max_{\vec{\lambda}}\{\log Z[\vec{\lambda}] - \vec{\lambda} \cdot \vec{\psi}$ where $\vec{\psi} = (1/N) \sum_{k=1}^K \vec{\phi}(\vec{x}^k)$ is the training data? Or, find $\vec{\lambda}^*$ such that model expectation of the statistics are equal to the data statistics? Mathematically:

$$\sum_{\vec{x}} \vec{\phi}(\vec{x}) P(\vec{x}|\vec{\lambda}^*) = \frac{1}{K} \sum_{k=1}^K \vec{\phi}(\vec{x}^k) \tag{5}$$

The answer is that these two approaches are exactly equivalent. To see this, we need to re-express the probability distribution for the Observable Markov Model in the form of an exponential distribution. We can re-write it as:

$$P(Q) = \frac{1}{Z[\vec{\lambda}^\pi, \vec{\lambda}^A]} \exp\{\vec{\lambda}^\pi \cdot \vec{\phi}^\pi(q_1) + \sum_{i=1}^{T-1} \vec{\lambda}^A \cdot \vec{\phi}^A(q_{i+1}, q_i)\}, \tag{6}$$

where the exponent terms are of form:

$$\vec{\lambda}^\pi \cdot \vec{\phi}^\pi(q_i) = \sum_{i=1}^N I(q_1 = s_i) \log \pi_i,$$
$$\vec{\lambda}^A \cdot \vec{\phi}^A(q_i, q_{i+1}) = \sum I(q_t = s_i, \text{ AND }, q_{t+1} = s_j) \log \overline{a}_{ij}. \tag{7}$$

Here $I(q = s)$ is the indicator function $I(q = s) = 1$ if $q = s$ and $I(q = s) = 0$ if $q \neq s$. NOTE $\overline{a}_{ij}$ is not exactly a_{ij} – CLARIFY THIS FOR NEXT DRAFT!!

The partition function is given by:

$$Z[\vec{\lambda}^\pi, \vec{\lambda}^A] = \sum_Q \exp\{\vec{\lambda}^\pi \cdot \vec{\phi}^\pi(q_1) + \sum_{i=1}^{T-1} \vec{\lambda}^A \cdot \vec{\phi}^A(q_{i+1}, q_i)\}. \tag{8}$$

It follows that the equations $\sum_{\vec{x}} \vec{\phi}(\vec{x}) P(\vec{x}|\vec{\lambda}) = \vec{\psi}$ can be solved analytically in this case to give the solutions in equations (4).

3 Hidden Markov Models

Now suppose that the states q are not directly observable. Instead for each state $q_t \in |\{s_1, ..., s_N\})$ we have an observable $O_t \in \{v_1, ..., v_M\}$. There is an observation probability $b_j(m) = P(O_t = v_m|q_t = s_j)$ that we observe v_m if we are in state s_j. (For example, the states are "biased coin" and "unbiased coin" the observables are "heads" or "tails". The probability of the observable being "heads" will depend on which coin is used).

This gives a full model with the following elements:

1. N: Number of states $S = \{s_1, ..., s_N\}$
2. M: Number of observation symbols $V = \{v_1, ..., v_M\}$
3. State transition probabilities: $A = \{a_{ij}\}$, with $a_{ij} P(q_{t+1} = s_j|q_t = s_i)$
4. Observation probabilities: $B = \{b_j(m)\}$, with $b_j(m) = P(O_t = v_m|q_t = s_j)$

5. Initial state probabilities: $\pi = \{\pi_i\}$, with $\pi_i = P(q_i = s_i)$.

There are three basic tasks that we will want HMMs to address:

1. Given a model $\lambda = (A, B, \pi)$, evaluate the probability $P(O|\lambda)$ of a sequence $O = (O_1, ..., O_T)$ (so that we can do model selection – use log-likelihood test to estimate whether a sequence of coin tosses it more likely to come from a fair coin or a biased coin).
2. Given a model λ and observation sequence $O = (O_1, ..., O_T)$, find the most probable states $Q = \{q_1, q_2, ..., q_T\}$ which has the highest probability of generating O: $Q^* = \arg\max_Q P(Q|O, \lambda)$.
3. Given a training set of sequences $X = \{O^k : k = 1, ..., K\}$ find the best values of the model parameters $\lambda^* = \arg\max_\lambda P(X|\lambda)$.

Observe that we specify the distribution $P(O, Q|\lambda) = P(O|Q, B)P(Q|A, \pi)$, where $P(O|Q, B) = \prod_{t=1}^T P(O_t = v_m|q_t = s_j) = \prod_{t=1}^T b_j(m)$ and $P(Q|A, \pi) = P(q_1|s_i) \prod_{t=1}^{T-1} P(q_{t+1} = s_j|q_t = s_k) = \pi_{q_1} \prod_{t=1}^{T-1} a_{q_t q_{t+1}}$.

This can also be expressed as exponential distribution with hidden variables – i.e., of the form $P(\vec{x}, \vec{y}|\vec{\lambda}) = (1/Z[\vec{\lambda}]) \exp\{\vec{\lambda} \cdot \vec{\phi}(\vec{x}, \vec{y})\}$ where $\vec{x}$ is observed and $\vec{y}$ is hidden. This requires an EM algorithm to solve for $\vec{\lambda}$ in order to learn the parameters $\vec{\lambda}$ from training data $\{\vec{x}^k : k = 1, ..., K\}$ (see last lecture).

The exponential form for an HMM is obtained by combining the prior distribution for the state, see equations (6,7), with a distribution for generating the data $P(O|Q)$. This is a factorized distribution which is of form:

$$P(O|Q) = \exp\{\vec{\lambda}^D \cdot \vec{\phi}^D(O, Q)\} = \exp\{\sum_{t=1}^T \sum i = 1^N \sum_{m=1}^M \log b_i(m) I(q_t = s_i, \text{ AND } O_t = v_m)\}, \quad (9)$$

where the normalization factor is 1 (because the distribution is factorized so it is trivial to do the normalization).

Putting everything together gives the distribution:

$$P(O, Q|\vec{\lambda}) = \frac{1}{Z[\vec{\lambda}^\pi, \vec{\lambda}^A]} \exp\{\vec{\lambda}^\pi \cdot \vec{\phi}(q_1) + \vec{\lambda}^A \cdot \vec{\phi}^A(Q) + \vec{\lambda}^D \cdot \vec{\phi}^D(O, Q)\}. \quad (10)$$

4 Task 1: Evaluation

We want to evaluate $P(O|\lambda)$ and can express this as:

$$P(O|\lambda) = \sum_Q P(O, Q|\lambda) = \sum_Q \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) b_{q_2}(O_2) ... a_{q_{T-1} q_T} b_{q_T}(O_T). \quad (11)$$

The problem is that Q has an exponential number of states N^T. Summing over this is impractical in general. Fortunately the form of the HMM makes this possible in polynomial time.

Define the *forward variable*:

$$\alpha_t(i) = P(O_1, ..., O_t, q_t = s_i|\lambda), \quad (12)$$

is the probability of generating all the observations up to time t and being in state q_t at time t (because of the Markov property – this can be computed independently of the observations after t.

We compute the forward variable recursively:

$$\alpha_1(i) = P(O_1, q_1 = s_i|\lambda) = \pi_i b_i(O_1)$$
$$\alpha_{t+1}(j) = \{\sum_{i=1}^N \alpha_t(i) a_{ij}\} b_j(O_{t+1}), \quad (13)$$

which enables us to compute $\alpha_T(i)$ is time $O(N^2 T)$.

After computing the α's, we can compute the probability of the data by:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i). \tag{14}$$

An alternative algorithm which can be used instead (and which we will need later for learning) is the *backward variable*:

$$\beta_t(i) = P(O_{t+1}, ..., O_T | q_t = s_i, \lambda), \tag{15}$$

which can be computed recursively by:

$$\beta_T(i) = 1,$$

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j). \tag{16}$$

This uses the Markov property that this is independent of the observations before t. We can also compute $\beta_1(i)$ in $O(N^2T)$ times and compute $P(O|\lambda) = \sum_{i=1}^{n} \beta_1(i)$.

Hence dynamic programming (i.e., the forward and backward algorithm) evaluates the probability of the data in polynomial time exploiting the Markov property of the model (e.g. the independence between the conditional probabilities of the early and late observations).

5 Task 2: Estimating the Best State Sequence

We can use DP to rapidly compute different properties of the state variables. For example, let $\gamma_t(i) = P(q_t = s_i|O, \lambda)$ be the marginal posterior of the t^{th} state.

Then we can easily compute

$$\gamma_t(i) = \frac{P(O|q_t = s_i, \lambda) P(q_t = s_i|\lambda)}{P(O|\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}. \tag{17}$$

NOTE; THE POSTERIOR MARGINAL IS SOMETIMES USED AS AN ALTERNATIVE TO THE MAP ESTIMATE (MOTIVATE FROM DECISION THEORY!!).

Often we want to estimate the MAP:

$$Q^* = \arg\max_{Q} P(Q|O, \lambda). \tag{18}$$

This can be done by the Viterbi algorithm (a form of DP) as follows. Define:

$$\delta_t(i) = \max_{q_1, ..., q_{t-1}} P(q_1, q_2, ..., q_{t-1}, q_t = s_i, O_1, ..., O_t|\lambda), \tag{19}$$

which is the probability of the highest probability path that accounts for all the first t observations and ends in state $q_t = s_i$.

We calculate $\delta_t(i)$ recursively by:

$$\text{Initialize } \delta_1(i) = \pi_i b_i(O_1), \ \psi_1(i) = 0 \tag{20}$$

$$\text{Recursion } \delta_t(j) = \max_i \delta_{t-1}(i) a_{ij} b_j(O_t)$$

$$\psi_t(j) = \arg\max_i \delta_{t-1}(i) a_{ij} \tag{21}$$

$$\text{Termination } p^* = \max_i \delta_T(i),$$

$$q_T^* = \arg\max_i \delta_T(i). \tag{22}$$

The best path Q^* can be found by backtracking: $q_t^* = \psi_{t+1}(q_{t+1}^*), \ t = T-1, T-2, ..., 1$.

The term $\psi_t(j)$ keeps track of the state that maximizes $\delta_t(j)$ at time $t-1$.

This algorithm also has complexity $O(N^2T)$ and so is efficient and practical.

6 Task 3: Learning Model Parameters

Let $X = \{O^k : k = 1, ..., K\}$ be a set of training sequences. We want to estimate the parameters λ by maximum likelihood:

$$\lambda^* = \arg\max_\lambda P(X|\lambda) = \arg\max_\lambda \prod_{k=1}^{K} P(O^k|\lambda). \tag{23}$$

This is performed by the EM algorithm using DP to make the computations practical.

Define:

$$\begin{aligned} \zeta_t(i,j) &= P(q_t = s_i, q_{t+1} = s_j|O, \lambda) \\ &= \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{k=1}^{N}\sum_{l=1}^{N}\alpha_t(k)a_{kl}b_l(O_{t+1})\beta_{t+1}(l)}. \end{aligned} \tag{24}$$

EM is performed by the Baum-Welch algorithm:

E-step: compute $\zeta_t(i,j)$ and $\gamma_t(i)$ using current estimate of λ.

M-step: recalculate λ form $\zeta_t(i,j)$ and $\gamma_t(i)$.

Recalculating λ gives:

$$\begin{aligned} a^*_{ij} &= \frac{\sum_{t=1}^{T-1}\zeta_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)}, \\ b^*_j(m) &= \frac{\sum_{t=1}^{T}\gamma_t(j)I(O_T = v_m)}{\sum_{t=1}^{T} I(O_T = v_m)}. \end{aligned} \tag{25}$$

For multiple sequences $X = \{O^k : k = 1, ..., K\}$ we have $P(X|\lambda) = \prod_{k=1}^{K} P(O^k|\lambda)$. This modifies the updates to:

$$\begin{aligned} a^*_{ij} &= \frac{\sum_{k=1}^{K}\sum_{t=1}^{T_1}\zeta_t(i,j)}{\sum_{k=1}^{K}\sum_{t=1}^{T-1}\gamma_t(i)}, \\ b^*_j(m) &= \frac{\sum_{k=1}^{K}\sum_{t=1}^{T}\gamma_t(j)I(O_T = v_m)}{\sum_{k=1}^{K}\sum_{t=1}^{T} I(O_T = v_m)}. \end{aligned} \tag{26}$$

CHECK FORMULAE FOR $b_j(m)$!!

$$\pi^*_i = \frac{\sum_{k=1}^{K}\gamma^k_1(i)}{K}. \tag{27}$$

This is the same as using the general form for EM with exponential distributions (see last lecture) and substituting in the exponential form of the HMM. Recall that this can be expressed as:

$$\begin{aligned} \mathrm{E-Step} \quad & q^{t+1}_\mu(\vec{y}_\mu) = P(\vec{y}_\mu|\vec{x}_\mu, \vec{\lambda}^t), \\ \mathrm{M-step\ \ solve\ for\ \ } \vec{\lambda}^{t+1} \text{ s.t. } & \sum_{\vec{y},\vec{x}} \vec{\phi}(\vec{x},\vec{y})P(\vec{x},\vec{y}|\vec{\lambda}^{t+1}) = \sum_\mu \sum_{\vec{y}_\mu} q^{t+1}_\mu(\vec{y}_\mu)\vec{\lambda}\cdot\vec{\phi}(\vec{x}_\mu,\vec{y}_\mu). \end{aligned} \tag{28}$$

Hence the M-step involves selecting the parameters $\vec{\lambda}$ so that the expected statistics of the model are equal to the observed statistics (averaged over the training set) and with the hidden states averaged with respect to the estimated distributions over the hidden states.