Hidden Markov models

Charles Elkan

November 26, 2012

Important: These lecture notes are based on notes written by Lawrence Saul. Also, these typeset notes lack illustrations. See the classroom lectures for figures and diagrams.

1 Simple Markov models

Bayesian networks are useful for modeling systems whose state changes over time. Here, the random variables are indexed by discrete time. That is, time t is an integer subscript on the names of the random variables, which can be called S_1, S_2, S_3, and so on. There may be a maximum time T, or there may be a countably infinite number of different states. An implicit assumption is that time t is discrete, not continuous. As for Bayesian networks in general, each random variable S_t may have discrete or continuous values. The set of possible state values is called the state space.

In a Bayesian network for modeling sequences s_t, the configuration of the edges (i.e., arrows) represents what simplifying assumptions we make. To make any progress, assumptions of some sort are needed. The most common assumption is called the Markov assumption, named after Andrey Markov (1856 to 1922). It is that each state S_t depends only on the previous state S_{t-1}. In other words, the earlier history S_1 to S_{t-2} is irrelevant. The network is $S_1 \to S_2 \to S_3 \to \cdots$.

Another fundamental assumption that is standard is that one CPT is shared across time: $p(S_v = s'|S_{v-1} = s) = p(S_u = s'|S_{u-1} = s)$ for all times u and v. This assumption is usually called stationarity. It is a special case of parameters being tied.

2 Hidden Markov models

Simple Markov models have at least two major weaknesses. First, taking into account dependence on k states in the past, that is making an order k Markov assumption, requires CPTs with an exponential number $O(n^k)$ of entries. Here, n is the cardinality of the state space, that is the number of alternative discrete states that the system may be in. Second, the model assumes that the state of the system can be observed directly. However, in many application domains, the true state is hidden, but at each time step, an observation $O_t = o_t$ is visible.

So, let O_t denote the observed data at time t, and S_t denote the state of the system at time t. In a hidden Markov model (HMM), the states S_t are never observed. They must be inferred from the observations O_t. For example, in robotics, observations O_t may be noisy sensor readings, while a state S_t may be the location and orientation of the robot.

HMMs are the standard approach to speech recognition. Intuitively, an observation O_t is a short window, maybe 20 milliseconds, of audio. A state S_t is a phoneme such as "ba." Note that the observations are real-valued while the states are discrete. This is easier to deal with than having real-valued states, but still allows working with the real-valued real world.

The Bayesian network for an HMM is an infinite polytree (draw the picture). The first-order Markov assumption makes the graph of the tree simple. The stationarity assumption makes the number of parameters constant. HMMs are a step towards overcoming both weaknesses of simple Markov models described above. Obviously, true states are not assumed to be observed. Less obviously, although the first order Markov property is true about the states, it is not true about the observations. That is, S_t is independent of S_{t-2} given S_{t-1}, but O_t is not independent of O_{t-2} given O_{t-1}. Indeed, every observation provides information about every other observation.

The joint distribution of the HMM is

$$p(S_1, S_2, \dots, S_T, O_1, O_2, \dots, O_T) = p(S_1) \cdot (\prod_{t=2}^{T} p(S_t|S_{t-1})) \cdot (\prod_{t=1}^{T} p(O_t|S_t)).$$

Let the set of alternative state values be $\{1, 2, \dots, n\}$. The parameters of the model are the initial state distribution $p(S_1 = i) = \pi_i$, the transition matrix $p(S_t = j|S_{t-1} = i) = a_{ij}$, and the emission matrix $p(O_t = k|S_t = i) = b_{ik}$. Note that i and j are members of the same set, but k is a member of a different set, namely the set of possible observations. Especially when k may be real-valued, sometimes we will write $b_i(k) = b_{ik}$.

There are two key computational questions for HMMs for which efficient algorithms are needed:

1. How to compute the most likely sequence of hidden state values? This sequence is $\text{argmax}_{\bar{s}}\ p(S_1, S_2, \ldots, S_T | O_1, O_2, \ldots, O_T)$. This computation is at the core of speech recognition.

2. How to obtain parameter values π_i, a_{ij}, b_{ik} that maximize the likelihood of observed data $p(O_1, O_2, \ldots, O_T)$?

The first problem above is inference, while the second is learning.

A critical subroutine in both problems is, given parameter values, to compute the likelihood $p(O_1, O_2, \ldots, O_T)$ of a given sequence of observations. We consider this task first.

End of the lecture on Tuesday November 13.

Start by marginalizing over the last state:

$$p(O_1, O_2, \ldots, O_T) = \sum_{j=1}^{n} p(O_1, O_2, \ldots, O_T, S_T = j).$$

Now, consider evaluating $p(O_1, O_2, \ldots, O_{t+1}, S_{t+1} = j)$ for any t. By summing over the possible values of S_i, applying the product rule for probabilities, and using the structure of the Bayesian network, this is

$$\sum_{i=1}^{n} p(O_1, O_2, \ldots, O_t, S_t = i) p(S_{t+1} = j | S_t = i) p(O_{t+1} | S_{t+1} = j).$$

In this sum, the first factor can be computed recursively, and the other two factors are entries in the CPTs of the HMM. In abbreviated notation,

$$\begin{aligned} \alpha_{it} &= p(O_1, O_2, \ldots, O_t, S_t = i) \\ \alpha_{j,t+1} &= \sum_{i=1}^{n} \alpha_{it} a_{ij} b_j(O_{t+1}). \end{aligned}$$

Here α is an n by T matrix whose entries are computed recursively, column by column, by what is called the forward algorithm. Each entry involves summing over the previous column, so the total time complexity is $O(n^2T)$. The base case of the recursion, and the leftmost column of the matrix, is

$$\alpha_{i1} = p(O_1, S_1 = i) = p(S_1 = i)p(O_1|S_1 = i) = \pi_i b_i(O_1).$$

The likelihood of an entire sequence of observations is

$$p(O_1, O_2, \ldots, O_T) = \sum_{j=1}^{n} \alpha_{jT}.$$

A word of warning: For long sequences, this probability can be less than 10^{-300}, which means it underflows numerically to zero in standard computer arithmetic. Instead of multiplying probabilities, it is better to add log probabilities.

Computing the most likely state sequence is a bit more difficult. This is

$$\begin{array}{rl} & \text{argmax}_{\bar{s}}\ p(S_1, S_2, \ldots, S_T|O_1, O_2, \ldots, O_T) \\ = & \text{argmax}_{\bar{s}}\ p(S_1, S_2, \ldots, S_T, O_1, O_2, \ldots, O_T)/p(O_1, O_2, \ldots, O_T) \\ = & \text{argmax}_{\bar{s}}\ \log p(S_1, S_2, \ldots, S_T, O_1, O_2, \ldots, O_T). \end{array}$$

Define ℓ_{it} to be the log probability of observations O_1 to O_t together with the most likely sequence of states that ends with state $S_t = i$ at time t. Formally,

$$\ell_{it} = \log \max_{S_1,\ldots,S_{t-1}} p(S_1, \ldots, S_{t-1}, S_t = i, O_1, \ldots, O_t).$$

We can form a recursion similar to the α_{it} recursion. The base case, for $t = 1$, is

$$\ell_{i1} = \log p(S_1 = i, O_1) = \log[p(S_1 = i)p(O_1|S_1 = i)] = \log \pi_i + \log b_i(O_1).$$

The recursive case goes from time t to time $t + 1$ and is

$$\begin{array}{rl} \ell_{j,t+1} = \max\limits_{S_1,\ldots,S_{t-1}} \max\limits_{i} & [\log p(S_1, \ldots, S_{t-1}, S_t = i, O_1, \ldots, O_t) \\ & \cdot p(S_{t+1} = j|S_t = i) \cdot p(O_{t+1}|S_{t+1} = j)] \end{array}$$

where the maximization over i is over values of S_t. Taking the logarithm of each factor separately and swapping the order of the maximizations yields

$$\ell_{j,t+1} = \max_i\ [\ell_{it} + \log a_{ij}] + \log b_j(O_{t+1}).$$

Note the similarity to the recursion for computing α_{ij}. The reason for the similarity is that a maximization operator has essentially the same algebraic properties as a summation operator.

The log probability of the most likely sequence is $\max_j \ell_{jT}$. A remaining issue is how to obtain the most likely state sequence, that is the argmax, from this maximum value. Switching in this way from the max to the argmax is a programming problem, not a mathematical one. Consider the n by T matrix whose entries are the ℓ_{it} values. The optimal last state, for time T, is whichever i gives the highest value in the Tth column. This is $S_T = \text{argmax}_j\ \ell_{jT}$. Now, consider finding the optimal second-to-last state, S_{T-1}. This is the state j that gives the highest value in the second-to-last column, when added to the log transition probability from j to S_T. Formally,

$$S_{T-1} = \text{argmax}_j\ [\ell_{j,T-1} + \log a_{j,S_T}].$$

Similar reasoning gives

$$S_{t-1} = \text{argmax}_j\ [\ell_{j,t-1} + \log a_{j,S_t}]$$

for each time t. In summary, the optimal path can be found from right to left after finding the log probability of the optimal path. Computing the entries of the log probability matrix requires $O(n^2T)$ time. After this matrix has been computed, then figuring out the actual optimal state sequence S_1 to S_T requires $O(nT)$ time.

End of the lecture on Thursday November 15.

The algorithm just described to find the optimal state sequence is called the Viterbi algorithm, after its inventor.[1] It is arguably the most important single algorithm in machine learning and signal processing. It is a special case of the general idea called dynamic programming.

3 EM training of HMMs

The goal of training is to estimate the parameters π_i, a_{ij}, and b_{ik}. The training data are a set of sequences each of the form $\bar{o} = o_1, o_2, \ldots, o_T$ where the length T may vary. The principle of maximum likelihood says that the aim is to maximize the probability of the training examples. For simplicity, we will assume there is just one training sequence.

[1]Andrew Viterbi, born 1935, Ph.D. in 1963 from USC, faculty at UCLA from 1963 to 1973, part-time faculty at UCSD from 1973 to 1994. Invented the Viterbi algorithm in 1967, cofounded Linkabit with Irwin Jacobs in 1968, also cofounded Qualcomm with Irwin Jacobs in 1985.

Because the outcomes of the state random variables are never observed, we will use expectation-maximization (EM). Using the general form of the M step for a Bayesian network, the M step updates are as follows:

$$\begin{aligned}\pi_i &:= p(S_1 = i|\bar{o}) \\ a_{ij} &:= \frac{\sum_t p(S_t = i, S_{t+1} = j|\bar{o})}{\sum_t p(S_t = i|\bar{o})} \\ b_{ik} &:= \frac{\sum_t p(S_t = i, O_t = k|\bar{o})}{\sum_t p(S_t = i|\bar{o})}.\end{aligned}$$

The E step is to compute each of the probabilities used above. Simple rules of probability and a results from above give

$$\begin{aligned}p(S_t = i, O_t = k|\bar{o}) &= p(S_t = i|\bar{o})I(o_t = k) \\ p(S_t = i|\bar{o}) &= \sum_{j=1}^{n} p(S_t = i, S_{t+1} = j|\bar{o}) \\ p(S_t = i, S_{t+1} = j|\bar{o}) &= p(S_t = i, S_{t+1} = j, \bar{o})/p(\bar{o}) \\ p(\bar{o}) &= \sum_{i=1}^{n} \alpha_{iT}\end{aligned}$$

There is just one difficult probability to compute:

$$\begin{aligned}p(S_t = i, S_{t+1} = j, \bar{o}) &= p(o_1, \ldots, o_t, S_t = i)p(S_{t+1} = j|o_1, \ldots, o_t, S_t = i) \\ &\quad \cdot p(o_{t+1}|S_{t+1} = j, o_1, \ldots, o_t, S_t = i) \\ &\quad \cdot p(o_{t+2}, \ldots, o_T|o_{t+1}, S_{t+1} = j, o_1, \ldots, o_t, S_t = i) \\ &= p(o_1, \ldots, o_t, S_t = i)p(S_{t+1} = j|S_t = i) \\ &\quad \cdot p(o_{t+1}|S_{t+1} = j) \cdot p(o_{t+2}, \ldots, o_T|S_{t+1} = j) \\ &= \alpha_{it} \cdot a_{ij} \cdot b_j(o_{t+1}) \cdot p(o_{t+2}, \ldots, o_T|S_{t+1} = j).\end{aligned}$$

By analogy to the definition $\alpha_{it} = p(o_1, \ldots, o_t, S_t = i)$, make the definition that $\beta_{j,t+1} = p(o_{t+2}, \ldots, o_T|S_{t+1} = j)$. We shall evaluate this probability recursively:

$$\begin{aligned}\beta_{it} &= p(o_{t+1}, \ldots, o_T|S_t = i) \\ &= \sum_{j=1}^{n} p(o_{t+1}, \ldots, o_T, S_{t+1} = j|S_t = i)\end{aligned}$$

$$
\begin{aligned}
&= \sum_{j=1}^{n} p(o_{t+1}, \ldots, o_T | S_{t+1} = j, S_t = i) p(S_{t+1} = j | S_t = i) \\
&= \sum_{j=1}^{n} p(o_{t+1}, \ldots, o_T | S_{t+1} = j) a_{ij} \\
&= \sum_{j=1}^{n} p(o_{t+1} | S_{t+1} = j) p(o_{t+2}, \ldots, o_T | S_{t+1} = j, o_{t+1}) a_{ij} \\
&= \sum_{j=1}^{n} b_j(o_{t+1}) p(o_{t+2}, \ldots, o_T | S_{t+1} = j) a_{ij} \\
&= \sum_{j=1}^{n} b_j(o_{t+1}) \beta_{j,t+1} a_{ij}.
\end{aligned}
$$

Operationally, the β values are computed by filling in an n by T matrix from right to left, one column at a time. The value of entry is a sum over the column to the right. The whole matrix is computed in $O(n^2T)$ time, like the α matrix.

Putting the results above together gives all the probabilities needed in the M step of EM for learning the parameters of the HMM. In particular,

$$
p(S_t = i, S_{t+1} = j | \bar{o}) = \frac{\alpha_{it} a_{ij} b_j(o_{t+1}) \beta_{j,t+1}}{\sum_i \alpha_{iT}}.
$$

The dynamic programming approach to compute the β matrix is called the backward algorithm. The EM method to learn the HMM parameters which is called the Baum-Welch algorithm; was published in 1970. From the beginning, it has been known that the Baum-Welch algorithm suffers from the problem of local optima. Remarkably, after 38 years, an algorithm to learn globally optimal HMM parameters was published in 2008. This paper is *A Spectral Algorithm for Learning Hidden Markov Models* by Daniel Hsu, Sham Kakade, and Tong Zhang. Daniel Hsu earned his Ph.D. in our department in 2010, advised by Sanjoy Dasgupta.

4 Linear dynamical systems

The word "dynamical" is not used in ordinary English, but is common in science. A dynamical system is any process with a state that changes as time goes by. A hidden Markov model, as seen so far, is a discrete dynamical system. However, the same Bayesian network can describe a system where the state values s and observations x are continuous, and even high-dimensional.

For tractability, assume that for all times t the state distribution is Gaussian, as is the observation distribution. The way to understand a multivariate Gaussian $N(\bar{x};\mu,\Sigma)$ is that $\bar{X} = \bar{x}$ is a vector-valued random variable, the vector μ is the average value of $\bar{x}$,

$$\mu = \mathbb{E}[\bar{x}] = \int p(\bar{x})\bar{x}\mathrm{d}\bar{x}$$

and the $\alpha\beta$ entry of Σ is the expected value of a re-centered product:

$$\Sigma_{\alpha\beta} = \mathbb{E}[(\bar{x}-\mu)_\alpha(\bar{x}-\mu)_\beta].$$

Gaussian distributions have a key property that all their marginal and conditional distributions are Gaussian. Hence, in a Bayesian network where all nodes have Gaussian distributions, all conditional distributions are Gaussian.

In a linear dynamical system, the means and variances of the Gaussians change over time. Let s_t be the state vector at time t. The state transition function is parameterized by a matrix A as $p(s_t|s_{t-1}) \sim N(As_{t-1},\Sigma^s)$. This is often written as $s_t = As_{t-1} + w$ with $w \sim N(0,\Sigma^s)$. Similarly, the emission function is $p(x_t|s_t) \sim N(Bs_t,\Sigma^x)$.

A posterior distribution is one that is obtained by conditioning on fixed values for evidence nodes. A particular case of a posterior distribution is the estimated state at time t conditioned on all observations until this time, $p(s_t|x_1,x_2,\ldots x_t)$. Computing this distribution is called Kalman filtering. The idea is that, for example, x_1 to x_t are radar measurements of an aircraft and s_t is its location. Note that, like in an HMM, $p(s_t|x_1,x_2,\ldots x_t)$ is not the same as $p(s_t|s_{t-1})$.

The forward, backward, and Baum-Welch algorithms are all applicable to linear dynamical systems, because their derivation depends only on the conditional independence properties inherent in the Bayesian network structure of an HMM, which is the same for a linear dynamical system. For Kalman filtering, the result is an update rule

$$\mathbb{E}[s_t|x_1,x_2,\ldots x_t] = \mu_t = A\mu_{t-1} + K_t(x_t - BA\mu_{t-1}).$$

The expression inside parentheses is the error of the actual observation x_t seen at time t compared to the prediction $BA\mu_{t-1}$ based on the state μ_{t-1} estimated for time $t-1$. The gain matrix K_t adjusts the estimated state based on this error. There is a similar, but more complicated, update rule for the estimated variance of s_t. The gain matrix depends on this estimated variance, so it changes over time.