properties of the fish to infer its identity (salmon or sea bass). The same network could instead be used to infer the probability that a fish is thin, or dark in color, based on probabilities of the identity of the fish, time of year, and so on (Problem 42).

When the dependency relationships among the features used by a classifier are unknown, we generally proceed by taking the simplest assumption, i.e., that the features are conditionally independent given the category, i.e.,

$$p(\omega_k|\mathbf{x}) \propto \prod_{i=1}^{d} p(x_i|\omega_k). \tag{85}$$

NAIVE BAYES RULE

In practice, this so-called *naive Bayes rule* or *idiot Bayes rule* often works quite well in practice, and can be expressed by a very simple belief net (Problem 43).

In Example 3 our entire belief net consisted of $\mathbf{X}$, its parents and children, and we needed to update only the values on $\mathbf{X}$. In the more general case, where the network is large, there may be many nodes whose values are unknown. In that case we may have to visit nodes randomly and update the probabilites until the entire configuration of probabilities is stable. It can be shown that under weak conditions, this process will converge to consistent values of the variables throughout the entire network (Problem 44).

Belief nets have found increasing use in complicated problems such as medical diagnosis. Here the upper-most nodes (ones without their own parents) represent a fundamental biological agent such as the presence of a virus or bacteria. Intermediate nodes then describe diseases, such as flu or emphysema, and the lower-most nodes the symptoms, such as high temperature or coughing. A physician enters measured values into the net and finds the most likely disease or cause. Such networks can be used in a somewhat more sophisticated way, automatically computing which unknown variable (node) should be measured to best reveal the identity of the disease.

We will return in Chap. **??** to address the problem of learning in such belief net models.

3.10 Hidden Markov Models

While belief nets are a powerful method for representing the dependencies and independencies among variables, we turn now to the problem of representing a particular but extremely important dependencies. In problems that have an inherent temporality — that is, consist of a process that unfolds in time — we may have states at time t that are influenced directly by a state at $t-1$. Hidden Markov models (HMMs) have found greatest use in such problems, for instance speech recognition or gesture recognition. While the notation and description is aunavoidably more complicated than the simpler models considered up to this point, we stress that the same underlying ideas are exploited. Hidden Markov models have a number of parameters, whose values are set so as to best explain training patterns for the known category. Later, a test pattern is classified by the model that has the highest posterior probability, i.e., that best "explains" the test pattern.

3.10.1 First-order Markov models

We consider a sequence of states at successive times; the state at any time t is denoted $\omega(t)$. A particular sequence of length T is denoted by $\boldsymbol{\omega}^T = \{\omega(1), \omega(2), ..., \omega(T)\}$ as

for instance we might have $\boldsymbol{\omega}^6 = \{\omega_1, \omega_4, \omega_2, \omega_2, \omega_1, \omega_4\}$. Note that the system can revisit a state at different steps, and not every state need be visited.

Our model for the production of any sequence is described by *transition probabilities* $P(\omega_j(t+1)|\omega_i(t)) = a_{ij}$ — the time-independent probability of having state ω_j at step $t+1$ given that the state at time t was ω_i. There is no requirement that the transition probabilities be symmetric ($a_{ij} \neq a_{ji}$, in general) and a particular state may be visited in succession ($a_{ii} \neq 0$, in general), as illustrated in Fig. 3.9.

TRANSITION PROBABILITY

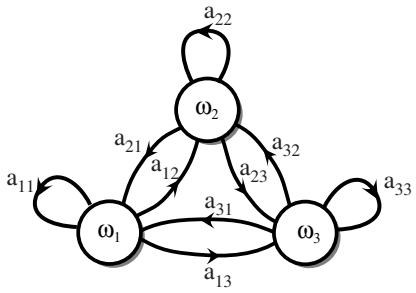


Figure 3.9: The discrete states, ω_i, in a basic Markov model are represented by nodes, and the transition probabilities, a_{ij}, by links. In a first-order discrete time Markov model, at any step t the full system is in a particular state $\omega(t)$. The state at step $t+1$ is a random function that depends solely on the state at step t and the transition probabilities.

Suppose we are given a particular model $\boldsymbol{\theta}$ — that is, the full set of a_{ij} — as well as a particular sequence $\boldsymbol{\omega}^T$. In order to calculate the probability that the model generated the particular sequence we simply multiply the successive probabilities. For instance, to find the probability that a particular model generated the sequence described above, we would have $P(\boldsymbol{\omega}^T|\boldsymbol{\theta}) = a_{14}a_{42}a_{22}a_{21}a_{14}$. If there is a prior probability on the first state $P(\omega(1) = \omega_i)$, we could include such a factor as well; for simplicity, we will ignore that detail for now.

Up to here we have been discussing a Markov model, or technically speaking, a *first-order* discrete time Markov model, since the probability at $t+1$ depends only on the states at t. For instance, in a Markov model for the production of spoken words, we might have states representing phonemes, and a Markov model for the production of a spoken work might have states representing phonemes. Such a Markov model for the word "cat" would have states for /k/, /a/ and /t/, with transitions from /k/ to /a/; transitions from /a/ to /t/; and transitions from /t/ to a final silent state.

Note however that in speech recognition the perceiver does not have access to the states $\omega(t)$. Instead, we measure some properties of the emitted sound. Thus we will have to augment our Markov model to allow for *visible states* — which are directly accessible to external measurement — as separate from the ω states, which are not.

3.10.2 First-order hidden Markov models

We continue to assume that at every time step t the system is in a state $\omega(t)$ but now we also assume that it emits some (visible) symbol $v(t)$. While sophisticated Markov models allow for the emission of continuous functions (e.g., spectra), we will restrict ourselves to the case where a discrete symbol is emitted. As with the states, we define

a particular sequence of such visible states as $\mathbf{V}^T = \{v(1), v(2), ..., v(T)\}$ and thus we might have $\mathbf{V}^6 = \{v_5, v_1, v_1, v_5, v_2, v_3\}$.

Our model is then that in any state $\omega(t)$ we have a probability of emitting a particular visible state $v_k(t)$. We denote this probability $P(v_k(t)|\omega_j(t)) = b_{jk}$. Because we have access only to the visible states, while the ω_i are unobservable, such a full model is called a *hidden Markov model* (Fig. 3.10)

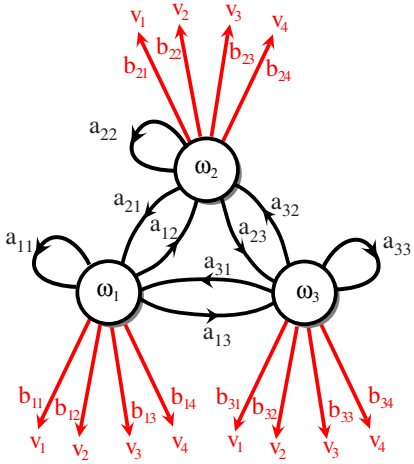


Figure 3.10: Three hidden units in an HMM and the transitions between them are shown in black while the visible states and the emission probabilities of visible states are shown in red. This model shows all transitions as being possible; in other HMMs, some such candidate transitions are not allowed.

3.10.3 Hidden Markov Model Computation

Now we define some new terms and clarify our notation. In general networks such as those in Fig. 3.10 are finite-state machines, and when they have associated transition probabilities, they are called Markov networks. They are strictly *causal* — the probabilities depend only upon *previous* states. A Markov model is called *ergodic* if every one of the states has a non-zero probability of occuring given some starting state. A

ABSORBING STATE

final or *absorbing state* ω_0 is one which, if entered, is never left (i.e., $a_{00} = 1$).

As mentioned, we denote the transition probabilities a_{ij} among hidden states and for the probability b_{jk} of the emission of a visible state:

$$\begin{aligned} a_{ij} &= P(\omega_j(t+1)|\omega_i(t)) \\ b_{jk} &= P(v_k(t)|\omega_j(t)). \end{aligned} \tag{86}$$

We demand that some transition occur from step $t \to t+1$ (even if it is to the same state), and that some visible symbol be emitted after every step. Thus we have the normalization conditions:

$$\sum_j a_{ij} = 1 \quad \text{for all } i \text{ and}$$

$$\sum_k b_{jk} = 1 \text{ for all } j, \tag{87}$$

where the limits on the summations are over all hidden states and all visible symbols, respectively.

With these preliminaries behind us, we can now focus on the three central issues in hidden Markov models:

The Evaluation problem. Suppose we have an HMM, complete with transition probabilites a_{ij} and b_{jk}. Determine the probability that a particular sequence of visible states $\mathbf{V}^T$ was generated by that model.

The Decoding problem. Suppose we have an HMM as well as a set of observations $\mathbf{V}^T$. Determine the most likely sequence of *hidden* states $\boldsymbol{\omega}^T$ that led to those observations.

The Learning problem. Suppose we are given the coarse structure of a model (the number of states and the number of visible states) but *not* the probabilities a_{ij} and b_{jk}. Given a set of training observations of visible symbols, determine these parameters.

We consider each of these problems in turn.

3.10.4 Evaluation

The probability that the model produces a sequence $\mathbf{V}^T$ of visible states is:

$$P(\mathbf{V}^T) = \sum_{r=1}^{r_{max}} P(\mathbf{V}^T|\boldsymbol{\omega}_r^T)P(\boldsymbol{\omega}_r^T), \tag{88}$$

where each r indexes a particular sequence $\boldsymbol{\omega}_r^T = \{\omega(1), \omega(2), ..., \omega(T)\}$ of T hidden states. In the general case of c hidden states, there will be $r_{max} = c^T$ possible terms in the sum of Eq. 88, corresponding to all possible sequences of length T. Thus, according to Eq. 88, in order to compute the probability that the model generated the particular sequence of T visible states $\mathbf{V}^T$, we should take each conceivable sequence of hidden states, calculate the probability they produce $\mathbf{V}^T$, and then add up these probabilities. The probability of a particular visible sequence is merely the product of the corresponding (hidden) transition probabilities a_{ij} and the (visible) output probabilities b_{jk} of each step.

Because we are dealing here with a first-order Markov process, the second factor in Eq. 88, which describes the transition probability for the hidden states, can be rewritten as:

$$P(\boldsymbol{\omega}_r^T) = \prod_{t=1}^{T} P(\omega(t)|\omega(t-1)) \tag{89}$$

that is, a product of the a_{ij}'s according to the hidden sequence in question. In Eq. 89, $\omega(T) = \omega_0$ is some final absorbing state, which uniquely emits the visible state v_0. In speech recognition applications, ω_0 typically represents a null state or lack of utterance, and v_0 is some symbol representing silence. Because of our assumption that the output probabilities depend only upon the hidden state, we can write the first factor in Eq. 88 as

$$P(\mathbf{V}^T|\boldsymbol{\omega}_r^T) = \prod_{t=1}^{T} P(v(t)|\omega(t)), \tag{90}$$

that is, a product of b_{jk}'s according to the hidden state and the corresponding visible state. We can now use Eqs. 89 & 90 to express Eq. 88 as

$$P(\mathbf{V}^T) = \sum_{r=1}^{r_{max}} \prod_{t=1}^{T} P(v(t)|\omega(t))P(\omega(t)|\omega(t-1)). \tag{91}$$

Despite its formal complexity, Eq. 91 has a straightforward interpretation. The probability that we observe the particular sequence of T visible states $\mathbf{V}^T$ is equal to the sum over all r_{max} possible sequences of hidden states of the conditional probability that the system has made a particular transition multiplied by the probability that it then emitted the visible symbol in our target sequence. All these are captured in our paramters a_{ij} and b_{kj}, and thus Eq. 91 can be evaluated directly. Alas, this is an $O(c^T T)$ calculation, which is quite prohibitive in practice. For instance, if $c = 10$ and $T = 20$, we must perform on the order of 10^{21} calculations.

A computationaly simpler algorithm for the same goal is as follows. We can calculate $P(\mathbf{V}^T)$ recursively, since each term $P(v(t)|\omega(t))P(\omega(t)|\omega(t-1))$ involves only $v(t)$, $\omega(t)$ and $\omega(t-1)$. We do this by defining

$$\alpha_i(t) = \begin{cases} 0 & t = 0 \text{ and } i \neq \text{initial state} \\ 1 & t = 0 \text{ and } i = \text{initial state} \\ \sum_j \alpha(t-1)a_{ij}b_{jk}v(t) & \text{otherwise,} \end{cases} \tag{92}$$

where the notation $b_{jk}v(t)$ means the transition probability b_{jk} selected by the visible state emitted at time t. thus the only non-zero contribution to the sum is for the index k which matches the visible state $v(t)$. Thus $\alpha_i(t)$ represents the probability that our HMM is in hidden state ω_i at step t having generated the first t elements of $\mathbf{V}^T$. This calculation is implemented in the *Forward algorithm* in the following way:

Algorithm 2 (HMM Forward)

1 **initialize** $\omega(1), t = 0, a_{ij}, b_{jk}$, visible sequence $\mathbf{V}^T, \alpha(0) = 1$
2 **for** $t \leftarrow t + 1$
3 $\alpha_j(t) \leftarrow \sum_{i=1}^{c} \alpha_i(t-1)a_{ij}b_{jk}$
4 **until** $t = T$
5 **return** $P(\mathbf{V}^T) \leftarrow \alpha_0(T)$
6 **end**

where in line 5, α_0 denotes the probability of the associated sequence ending to the known final state. The *Forward algorithm* has, thus, a computational complexity of $O(c^2T)$ — *far* more efficient than the complexity associated with exhaustive enumeration of paths of Eq. 91 (Fig. 3.11). For the illustration of $c = 10$, $T = 20$ above, we would need only on the order of 2000 calculations — more than 17 orders of magnitude faster than that to examine each path individually.

We shall have cause to use the *Backward algorithm*, which is the time-reversed version of the *Forward algorithm*.

Algorithm 3 (HMM Backward)

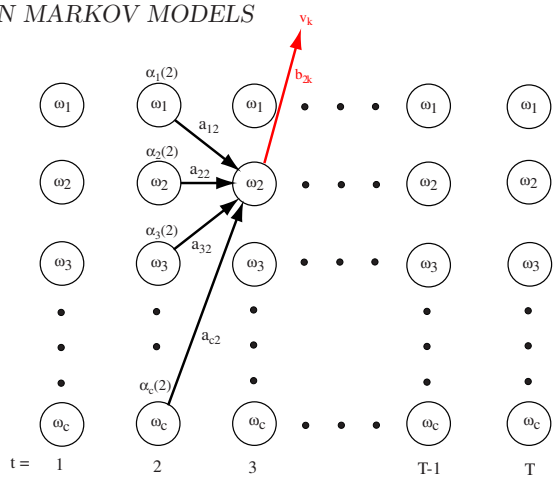


Figure 3.11: The computation of probabilities by the Forward algorithm can be visualized by means of a trellis — a sort of "unfolding" of the HMM through time. Suppose we seek the probability that the HMM was in state ω_2 at $t = 3$ and generated the observed visible up through that step (including the observed visible symbol v_k). The probability the HMM was in state $\omega_j(t = 2)$ and generated the observed sequence through $t = 2$ is $\alpha_j(2)$ for $j = 1, 2, ..., c$. To find $\alpha_2(3)$ we must sum these and multiply the probability that state ω_2 emitted the observed symbol v_k. Formally, for this particular illustration we have $\alpha_2(3) = b_{2k} \sum_{j=1}^{c} \alpha_j(2) a_{j2}$.

initialize $\omega(T), t = T, a_{ij}, b_{jk}$, visible sequence V^T
  for $t \leftarrow t - 1$;
    $\beta_j(t) \leftarrow \sum_{i=1}^{c} \beta_i(t+1) a_{ij} b_{jk} v(t+1)$
  until $t = 1$
return $P(V^T) \leftarrow \beta_i(0)$ for the known initial state
end

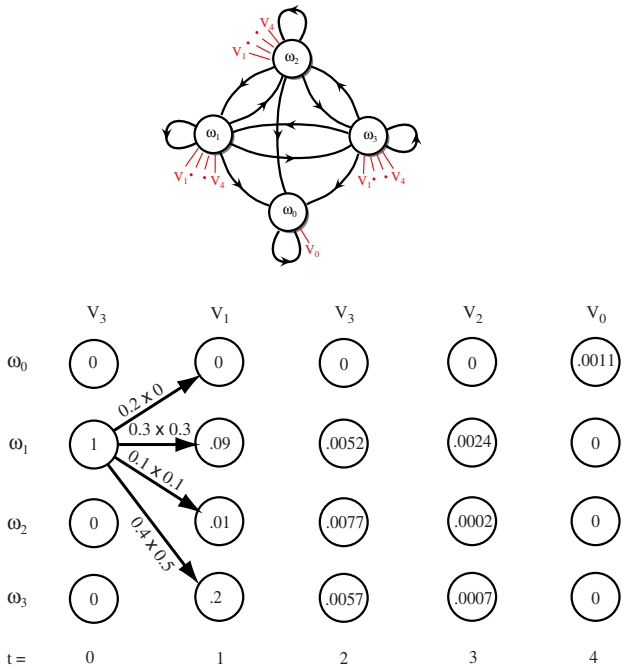
Example 4: Hidden Markov Model

To clarify the evaluation problem, consider an HMM such as shown in Fig. 3.10, but with an explicit absorber state and unique null visible symbol V_0 with the following transition probabilities (where the matrix indexes begin at 0):

$$a_{ij} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.2 & 0.3 & 0.1 & 0.4 \\ 0.2 & 0.5 & 0.2 & 0.1 \\ 0.8 & 0.1 & 0.0 & 0.1 \end{pmatrix} \text{ and}$$

$$b_{jk} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0.3 & 0.4 & 0.1 & 0.2 \\ 0 & 0.1 & 0.1 & 0.7 & 0.1 \\ 0 & 0.5 & 0.2 & 0.1 & 0.2 \end{pmatrix}.$$

What is the probability it generates the particular sequence $\mathbf{V}^5 = \{v_3, v_1, v_3, v_2, v_0\}$? Suppose we know the initial hidden state at $t = 0$ to be ω_1. The visible symbol at each step is shown above, and the $\alpha_i(t)$ in each unit. The circles show the value for $\alpha_i(t)$ as we progress left to right. The product $a_{ij}b_{jk}$ is shown along each transition link for the step $t = 1$ to $t = 2$. The final probability, $P(\mathbf{V}^T|\boldsymbol{\theta})$ is hence 0.0011.



The HMM (above) consists of four hidden states (one of which is an absorber state, ω_0), each emitting one of five visible states; only the allowable transitions to visible states are shown. The trellis for this HMM is shown below. In each node is $\alpha_i(t)$ — the probability the model generated the observed visible sequence up to t. For instance, we know that the system was in hidden state ω_1 at $t = 1$, and thus $\alpha_1(0) = 1$ and $\alpha_i(0) = 0$ for $i \neq 1$. The arrows show the calculation of $\alpha_i(1)$. for instance, since visible state v_1 was emitted at $t = 1$, we have $\alpha_0(1) = \alpha_1(0)a_{10}b_{01} = 1[0.2 \times 0] = 0$. as shown by the top arrow. Likewise the nest highest arrow corresponds to the calculation $\alpha_1(1) = \alpha_1(0)a_{11}b_{11} = 1[0.3 \times 0.3] = 0.09$. In this example, the calculation of $\alpha_i(1)$ is particularly simple, since only transitions from the known initial hidden state need be considered; all other transitions have zero contribution to $\alpha_i(1)$. For subsequent times, however, the caculation requires a *sum* over all hidden states at the previous time, as given by line 3 in the Forward algorithm. The probability shown in the final (absorbing) state gives the probability of the full sequence observed, $P(\mathbf{V}^T|\boldsymbol{\theta}) = 0.0011$.

If we denote our model — the a's and b's — by $\boldsymbol{\theta}$, we have by Bayes' formula that the probability of the model given the observed sequence is:

$$P(\boldsymbol{\theta}|\mathbf{V}^T) = \frac{P(\mathbf{V}^T|\boldsymbol{\theta})P(\boldsymbol{\theta})}{P(\mathbf{V}^T)} \tag{93}$$

In HMM pattern recognition we would have a number of HMMs, one for each category and classify a test sequence according to the model with the highest probability. Thus in HMM speech recognition we could have a model for "cat" and another one for "dog" and for a test utterance determine which model has the highest probability. In practice, nearly all HMMs for speech are *left-to-right* models (Fig. 3.12).

LEFT-TO-RIGHT MODEL

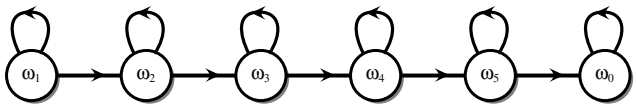


Figure 3.12: A left-to-right HMM commonly used in speech recognition. For instance, such a model could describe the utterance "viterbi," where ω_1 represents the phoneme /v/, ω_2 represents /i/, ..., and ω_0 a final silent state. Such a left-to-right model is more restrictive than the general HMM in Fig. 3.10, and precludes transitions "back" in time.

The `Forward algorithm` gives us $P(V^T|\boldsymbol{\theta})$. The prior probability of the model, $P(\boldsymbol{\theta})$, is given by some external source, such as a *language model* in the case of speech. This prior probability might depend upon the semantic context, or the previous words, or yet other information. In the absence of such information, it is traditional to assume a uniform density on $P(\boldsymbol{\theta})$, and hence ignore it in any classification problem. (This is an example of a "non-informative" prior.)

3.10.5 Decoding

Given a sequence of visible states $\mathbf{V}^T$, the decoding problem is to find the most probable sequence of hidden states. While we might consider enumerating every possible path and calculating the probability of the visible sequence observed, this is an $O(c^T T)$ calculation and prohibitive. Instead, we use perhaps the simplest decoding algorithm:

Algorithm 4 (HMM decoding)

```
1  begin initialize  Path = {}, t = 0
2        for  t ← t + 1
4            k = 0, α_0 = 0
5            for  k ← k + 1
7                α_k(t) ← b_jk v(t) Σ_{i=1}^{c} α_i(t − 1) a_ij
8            until k = c
10           j' ← arg max_j α_j(t)
11           AppendTo Path ω_j'
12       until t = T
13   return Path
14 end
```

A closely related algorithm uses logarithms of the probabilities and calculates total probabilities by addition of such logarithms; this method has complexity $O(c^2 T)$ (Problem 48).

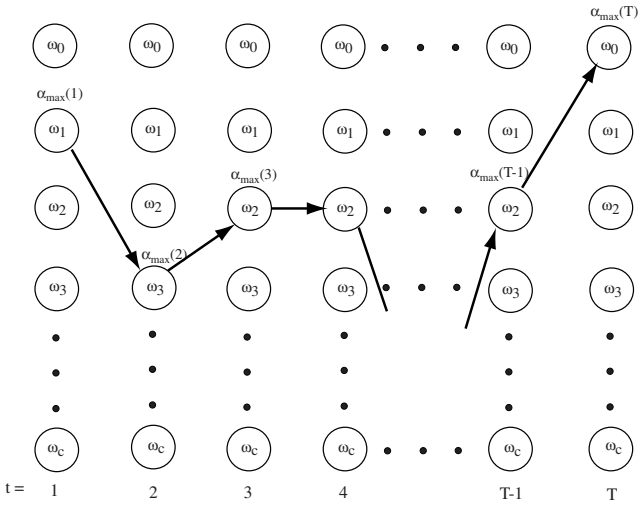
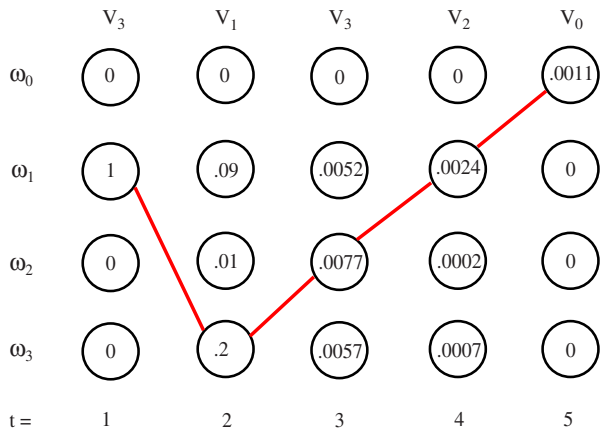


Figure 3.13: The decoding algorithm finds at each time step t the state that has the highest probability of having come from the previous step and generated the observed visible state v_k. The full path is the sequence of such states. Because this is a local optimization (dependent only upon the single previous time step, not the full sequence), the algorithm does not guarantee that the path is indeed allowable. For instance, it might be possible that the maximum at $t = 5$ is ω_1 and at $t = 6$ is ω_2, and thus these would appear in the path. This can even occur if $a_{12} = P(\omega_2(t+1)|\omega_1(t)) = 0$, precluding that transition.

The red line in Fig. 3.13 corresponds to Path, and connects the hidden states with the highest value of α_i at each step t. There is a difficulty, however. Note that there is no guarantee that the path is in fact a *valid* one — it might not be consistent with the underlying models. For instance, it is possible that the path actually implies a transition that is forbidden by the model, as illustrated in Example 5.

Example 5: HMM decoding

We find the path for the data of Example 4 for the sequence $\{\omega_1, \omega_3, \omega_2, \omega_1, \omega_0\}$. Note especially that the transition from ω_3 to ω_2 is not allowed according to the transition probabilities a_{ij} given in Example 4. The path *locally* optimizes the probability through the trellis.



The locally optimal path through the HMM trellis of Example 4.

HMMs address the problem of rate invariance in the following two ways. The first is that the transition probabilities themselves incorporate probabilistic structure of the durations. Moreover, using postprocessing, we can delete repeated states and just get the sequence somewhat independent of variations in rate. Thus in post-processing we can convert the sequence $\{\omega_1, \omega_1, \omega_3, \omega_2, \omega_2, \omega_2\}$ to $\{\omega_1, \omega_3, \omega_2\}$, which would be appropriate for speech recognition, where the fundamental phonetic units are not repeated in natural speech.

3.10.6 Learning

The goal in HMM learning is to determine model parameters — the transition probabilities a_{ij} and b_{jk} — from an ensemble of training samples. There is no known method for obtaining the optimal or most likely set of parameters from the data, but we can nearly always determine a good solution by a straightforward technique.

The Forward-backward Algorithm

The Forward-backward algorithm is an instance of a generalized Expectation-Maximization algorithm. The general approach will be to iteratively update the weights in order to better explain the observed training sequences.

Above, we defined $\alpha_i(t)$ as the probability that the model is in state $\omega_i(t)$ and has generated the target sequence up to step t. We can analogously define $\beta_i(t)$ to be the probability that the model is in state $\omega_i(t)$ and *will generate* the remainder of the given target sequence, i.e., from $t+1 \rightarrow T$. We express $\beta_i(t)$ as:

$$\beta_i(t) = \begin{cases} 0 & \omega_i(t) \neq \text{sequence's final state and } t = T \\ 1 & \omega_i(t) = \text{sequence's final state and } t = T \\ \sum_j a_{ij} b_{jk} v(t+1) \beta_j(t+1) & \text{otherwise,} \end{cases} \tag{94}$$

To understand Eq. 94, imagine we knew $\alpha_i(t)$ up to step $T-1$, and we wanted to calculate the probability that the model would generate the remaining single visible

symbol. This probability, $\beta_i(T)$, is just the probability we make a transition to state $\omega_i(T)$ multiplied by the probability that this hidden state emitted the correct final visible symbol. By the definition of $\beta_i(T)$ in Eq. 94, this will be either 0 (if $\omega_i(T)$ is not the final hidden state) or 1 (if it is). Thus it is clear that $\beta_i(T-1) = \sum_j a_{ij} b_{ij} v(T) \beta_i(T)$. Now that we have determined $\beta_i(T-1)$, we can repeat the process, to determine $\beta_i(T-2)$, and so on, *backward* through the trellis of Fig. **??**.

But the $\alpha_i(t)$ and $\beta_i(t)$ we determined are merely *estimates* of their true values, since we don't know the actual value of the transition probabilities a_{ij} and b_{ij} in Eq. 94. We can calculate an improved value by first defining $\gamma_{ij}(t)$ — the probability of transition between $\omega_i(t-1)$ and $\omega_j(t)$, given the model generated the entire training sequence $\mathbf{V}^T$ by *any* path. We do this by defining $\gamma_{ij}(t)$, as follows:

$$\gamma_{ij}(t) = \frac{\alpha_i(t-1) a_{ij} b_{ij} \beta_i(t)}{P(V^T|\boldsymbol{\theta})}, \tag{95}$$

where $P(\mathbf{V}^T|\boldsymbol{\theta})$ is the probability that the model generated sequence $\mathbf{V}^T$ by any path. Thus $\gamma_{ij}(t)$ is the probability of a transition from state $\omega_i(t-1)$ to $\omega_j(t)$ given that the model generated the complete visible sequence V^T.

We can now calculate an improved estimate for a_{ij}. The expected number of transitions between state $\omega_i(t-1)$ and $\omega_j(t)$ at *any* time in the sequence is simply $\sum_{t=1}^{T} \gamma_{ij}(t)$, whereas at step t it is $\sum_{t=1}^{T} \sum_k \gamma_{ik}(t)$. Thus $\hat{a}_{ij}$ (the estimate of the probability of a transition from $\omega_i(t-1)$ to $\omega_j(t)$) can be found by taking the ratio between the expected number of transitions from ω_i to ω_j and the total expected number of *any* transitions from ω_i. That is:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T} \gamma_{ij}(t)}{\sum_{t=1}^{T} \sum_k \gamma_{ik}(t)}. \tag{96}$$

In the same way, we can obtain an improved estimate $\hat{b}_{ij}$ by calculating the ratio between the frequency that any particular symbol v_k is emitted and that for any symbol. Thus we have

$$\hat{b}_{jk} = \frac{\sum \gamma_{jk}(t)}{\sum_{t=1}^{T} \gamma_{jk}(t)}. \tag{97}$$

In short, then, we start with rough or arbitrary estimates of a_{ij} and b_{jk}, calculate improved estimates by Eqs. 96 & 97, and repeat until some convergence criterion is met (e.g., sufficiently small change in the estimated values of the parameters on subsequent iterations). This is the *Baum-Welch* or *Forward-backward algorithm* — an example of a Generalized Expectation-Maximumization algorithm (Sec. 3.8):

Algorithm 5 (Forward-backward)

1 **begin initialize** a_{ij}, b_{jk}, training sequence V^T, convergence criterion θ
2 **do** $z \leftarrow z + 1$
3 Compute $\hat{a}(z)$ from $a(z-1)$ and $b(z-1)$ by Eq. 96
4 Compute $\hat{b}(z)$ from $a(z-1)$ and $b(z-1)$ by Eq. 97
5 $a_{ij}(z) \leftarrow \hat{a}_{ij}(z-1)$
6 $b_{jk}(z) \leftarrow \hat{b}_{jk}(z-1)$

7 **until** $\max_{i,j,k}[a_{ij}(z) - a_{ij}(z-1), b_{jk}(z) - b_{jk}(z-1)] < \theta$; convergence achieved$ln : ForBackstop$
8 **return** $a_{ij} \leftarrow a_{ij}(z)$; $b_{jk} \leftarrow b_{jk}(z)$
9 **end**

The stopping or convergence criterion in line **??** halts learning when no estimated transition probability changes more than a predetermined amount, θ. In typical speech recognition applications, convergence requires several presentations of each training sequence (fewer than five is common). Other popular stopping criteria are based on overall probability that the learned model could have generated the full training data.

Summary

If we know a parametric form of the class-conditional probability densities, we can reduce our learning task from one of finding the distribution itself, to that of finding the *parameters* (represented by a vector $\boldsymbol{\theta}_i$ for each category ω_i), and use the resulting distributions for classification. The maximum likelihood method seeks to find the parameter value that is best supported by the training data, i.e., maximizes the probability of obtaining the samples actually observed. (In practice, for computational simplicity one typically uses log-likelihood.) In Bayesian estimation the parameters are considered random variables having a known a priori density; the training data convert this to an a posteriori density. The recursive Bayes method updates the Bayesian parameter estimate incrementally, i.e., as each training point is sampled. While Bayesian estimation is, in principle, to be preferred, maximum likelihood methods are generally easier to implement and in the limit of large training sets give classifiers nearly as accurate.

A sufficient statistic $\mathbf{s}$ for $\boldsymbol{\theta}$ is a function of the samples that contains all information needed to determine $\boldsymbol{\theta}$. Once we know the sufficient statistic for models of a given form (e.g., exponential family), we need only estimate their value from data to create our classifier — no other functions of the data are relevant.

Expectation-Maximization is an iterative scheme to maximize model parameters, even when some data are missing. Each iteration employs two steps: the expectation or **E step** which requires marginalizing over the missing variables given the current model, and the maximization or **M step**, in which the optimum parameters of a new model are chosen. Generalized Expectation-Maximization algorithms demand merely that parameters be *improved* — not optimized — on each iteration and have been applied to the training of a large range of models.

Bayesian belief nets allow the designer to specify, by means of connection topology, the functional dependences and independencies among model variables. When any subset of variables is clamped to some known values, each node comes to a probability of its value through a Bayesian inference calculation. Parameters representing conditional dependences can be set by an expert.

Hidden Markov models consist of nodes representing hidden states, interconnected by links describing the conditional probabilities of a transition between the states. Each hidden state also has an associated set of probabilities of emiting a particular visible states. HMMs can be useful in modelling sequences, particularly context dependent ones, such as phonemes in speech. All the transition probabilities can be learned (estimated) iteratively from sample sequences by means of the *Forward-backward* or *Baum-Welch* algorithm, an example of a generalized EM algorithm. Classification

proceeds by finding the single model among candidates that is most likely to have produced a given observed sequence.

Bibliographical and Historical Remarks

Maximum likelihood and Bayes estimation have a long history. The Bayesian approach to learning in pattern recognition began by the suggestion that the proper way to use samples when the conditional densities are unknown is the calculation of $P(\omega_i|\mathbf{x}, \mathcal{D})$, [6]. Bayes himself appreciated the role of non-informative priors. An analysis of different priors from statistics appears in [21, 15] and [4] has an extensive list of references.

The origins of Bayesian belief nets traced back to [33], and a thorough literature review can be found in [8]; excellent modern books such as [24, 16] and tutorials [7] can be recommended. An important dissertation on the theory of belief nets, with an application to medical diagnosis is [14], and a summary of work on diagnosis of machine faults is [13]. While we have focussed on directed acyclic graphs, belief nets are of broader use, and even allow loops or arbitrary topologies — a topic that would lead us far afield here, but which is treated in [16].

The Expectation-Maximization algorithm is due to Dempster et al.[11] and a thorough overview and history appears in [23]. On-line or incremental versions of EM are described in [17, 31]. The definitive compendium of work on missing data, including much beyond our discussion here, is [27].

Markov developed what later became called the Markov framework [22] in order to analyze the the text of his fellow Russian Pushkin's masterpiece **Eugene Onegin**. Hidden Markov models were introduced by Baum and collaborators [2, 3], and have had their greatest applications in the speech recognition [25, 26], and to a lesser extent statistical language learning [9], and sequence identification, such as in DNA sequences [20, 1]. Hidden Markov methods have been extended to two-dimensions and applied to recognizing characters in optical document images [19]. The decoding algorithm is related to pioneering work of Viterbi and followers [32, 12]. The relationship between hidden Markov models and graphical models such as Bayesian belief nets is explored in [29].

Knuth's classic [18] was the earliest compendium of the central results on computational complexity, the majority due to himself. The standard books [10], which inspired several homework problems below, are a bit more accessible for those without deep backgrounds in computer science. Finally, several other pattern recognition textbooks, such as [28, 5, 30] which take a somewhat different approach to the field can be recommended.

Problems

⨁ *Section 3.2*

1. Let x have an exponential density

$$p(x|\theta) = \begin{cases} \theta e^{-\theta x} & x \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

(a) Plot $p(x|\theta)$ versus x for $\theta = 1$. Plot $p(x|\theta)$ versus θ, $(0 \leq \theta \leq 5)$, for $x = 2$.