6

HIDDEN MARKOV MODELS AND LOGLINEAR MODELS

In this chapter we introduce two important classes of statistical models for processing text and speech, the **Hidden Markov Model** (**HMM**) and the **Maximum Entropy** model (**MaxEnt**).

HMMs and MaxEnt are machine learning models. We have already touched on some aspects of machine learning; we briefly introduced the Hidden Markov Model in the previous chapter, and we have introduced the *N*-gram model in the chapter before. In this chapter we give a more complete introduction to such models, in preparation for the many statistical models that we will see throughout the book, including **Naive Bayes**, **decision lists**, and **Gaussian Mixture Models**.

6.1 MARKOV CHAINS

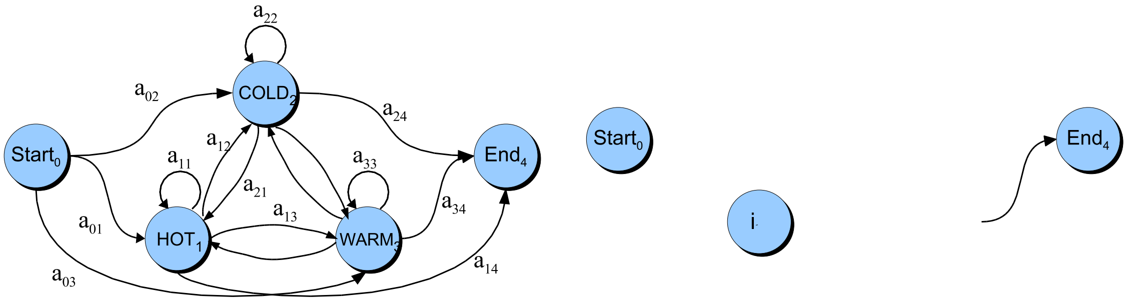
The Hidden Markov Model is one of the most important machine learning models in speech and language processing. In order to define it properly, we need to first introduce the **Markov chain**, sometimes called the **observed Markov model**. Markov chains and Hidden Markov Models are both extensions of the finite automata of Ch. 3. Recall that a finite automaton is defined by a set of states, and a set of transitions between states that are taken based on the input observations. A **weighted finite-state automaton** is a simple augmentation of the finite automaton in which each arc is associated with a probability, indicating how likely that path is to be taken. The probability on all the arcs leaving a node must sum to 1.

WEIGHTED

MARKOV CHAIN

A **Markov chain** is a special case of a weighted automaton in which the input sequence uniquely determines which states the automaton will go through. Because they can't represent inherently ambiguous problems, a Markov chain is only useful for assigning probabilities to unambiguous sequences.

Fig. 6.1a shows a Markov chain for assigning a probability to a sequence of weather events, where the vocabulary consists of HOT, COLD, and RAINY,. Fig. 6.1b shows another simple example of a Markov chain for assigning a prob-


a22
a02
COLD2
a24
Start0
a12
a11
a33
End4
a21
a13
a34
a01
HOT1
WARM3
a14
a03
Start0
End4
i

being in state j after seeing the first t observations, given the automaton λ. The value of each cell $\alpha_t(j)$ is computed by summing over the probabilities of every path that could lead us to this cell. Formally, each cell expresses the following probability:

$$\alpha_t(j) = P(o_1, o_2 \ldots o_t, q_t = j|\lambda) \tag{6.10}$$

Here $q_t = j$ means "the probability that the tth state in the sequence of states is state j". We compute this probability by summing over the extensions of all the paths that lead to the current cell. An extension of a path from a state i at time $t-1$ is computed by multiplying the following three factors:

1. the **previous path probability** from the previous cell $\alpha_{t-1}(i)$,
2. the **transition probability** a_{ij} from previous state i to current state j, and
3. the **state observation likelihood** $b_j(o_t)$ that current state j matches observation symbol t.

Consider the computation in Fig. 6.6 of $\alpha_2(1)$, the forward probability of being at time step 2 in state 1 having generated the partial observation *3 2*. This is computed by extending the α probabilities from time step 1, via two paths, each extension consisting of the three factors above: $\alpha_1(1) \times P(H|H) \times P(1|H)$ and $\alpha_1(2) \times P(H|C) \times P(1|H)$.

Fig. 6.7, adapted from Rabiner (1989), shows another visualization of this induction step for computing the value in one new cell of the trellis.

We give two formal definitions of the Forward algorithm; the pseudocode in Fig. 6.8 and a statement of the definitional recursion here:

1. Initialization:

$$\alpha_1(j) \;=\; a_{0j} b_j(o_1) \quad 1 \le j \le N \tag{6.11}$$

2. Recursion (since states 0 and N are non-emitting):

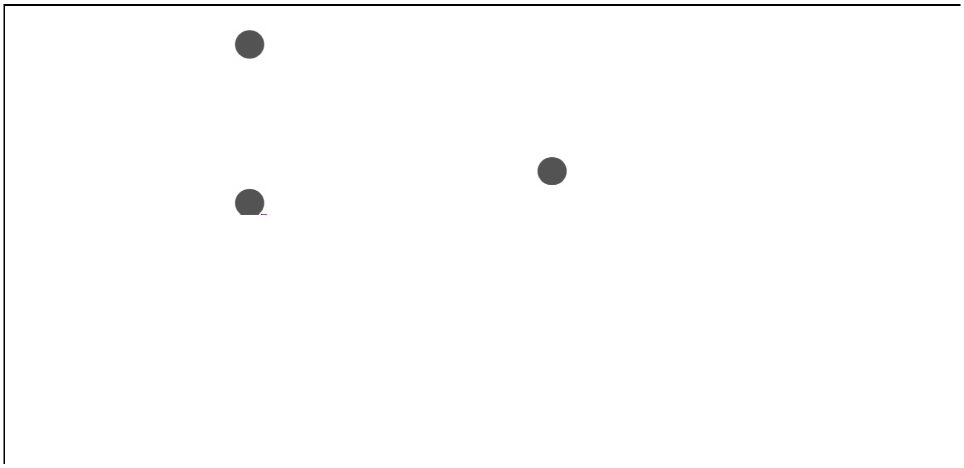
$$\alpha_t(j) = \left[\sum_{i=1}^{N-1} \alpha_{t-1}(i) a_{ij}\right] b_j(o_t); \quad 1 < j < N, 1 < t < T \tag{6.12}$$

3. Termination:

$$P(O|\lambda) = \alpha_T(N) = \sum_{i=2}^{N-1} \alpha_T(i) a_{iN} \tag{6.13}$$

6.4 DECODING: THE VITERBI ALGORITHM

For any model, such as an HMM, that contains hidden variables, the task of determining which sequence of variables is the underlying source of some sequence



are observed, we can run the model on the observation sequence and directly see which path we took through the model, and which state generated each observation symbol. A Markov chain of course has no emission probabilities B (alternatively we could view a Markov chain as a degenerate Hidden Markov Model where all the b probabilities are 1.0 for the observed symbol and 0 for all other symbols.). Thus the only probabilities we need to train are the transition probability matrix A.

We get the maximum likelihood estimate of the probability a_{ij} of a particular transition between states i and j by counting the number of times the transition was taken, which we could call $C(i \rightarrow j)$, and then normalizing by the total count of all times we took any transition from state i:

$$a_{ij} = \frac{C(i \rightarrow j)}{\sum_{q \in Q} C(i \rightarrow q)} \tag{6.21}$$

We can directly compute this probability in a Markov chain because we know which states we were in. For an HMM we cannot compute these counts directly from an observation sequence since we don't know which path of states was taken through the machine for a given input. The Baum-Welch algorithm uses two neat intuitions to solve this problem. The first idea is to *iteratively* estimate the counts. We will start with an estimate for the transition and observation probabilities, and then use these estimated probabilities to derive better and better probabilities. The second idea is that we get our estimated probabilities by computing the forward probability for an observation and then dividing that probability mass among all the different paths that contributed to this forward probability.

In order to understand the algorithm, we need to define a useful probability related to the forward probability, called the **backward probability**.

BACKWARD PROBABILITY

The backward probability β is the probability of seeing the observations from time $t+1$ to the end, given that we are in state j at time t (and of course given the automaton λ):

$$\beta_t(i) = P(o_{t+1}, o_{t+2} \ldots o_T | q_t = i, \lambda) \tag{6.22}$$

It is computed inductively in a similar manner to the forward algorithm.

1. Initialization:

$$\beta_T(i) \ = \ a_{iN}, \ 1 < i < N \tag{6.23}$$

2. Recursion (again since states 0 and N are non-emitting):

$$\beta_t(i) = \sum_{i=1}^{N-1} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \ 1 < i < N, 0 < t < T \tag{6.24}$$

3. Termination:

$$P(O|\lambda) = \alpha_T(N) = \beta_T(1) = \sum_{j=1}^{N-1} a_{1j} b_j(o_1) \beta_1(j) \tag{6.25}$$

Fig. 6.12 illustrates the backward induction step.

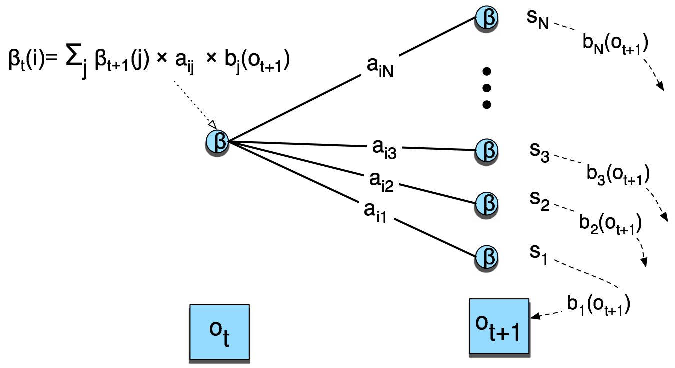


Figure 6.12 The computation of $\beta_t(i)$ by summing all the successive values $\beta_{t+1}(j)$ weighted by their transition probabilities a and their observation probabilities $b_j(o_{t+1})$. After (Rabiner, 1989)

We are now ready to understand how the forward and backward probabilities can help us compute the transition probability a_{ij} and observation probability $b_i(o_t)$ from an observation sequence, even though the actual path taken through the machine is hidden.

Let's begin by showing how to reestimate a_{ij}. We will proceed to estimate $\hat{a}_{ij}$ by a variant of (6.21):

$$\hat{a}_{ij} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i} \tag{6.26}$$

How do we compute the numerator? Here's the intuition. Assume we had some estimate of the probability that a given transition $i \rightarrow j$ was taken at a particular point in time t in the observation sequence. If we knew this probability for each particular time t, we could sum over all times t to estimate the total count for the transition $i \rightarrow j$.

More formally, let's define the probability ξ_t as the probability of being in state i at time t and state j at time $t+1$, given the observation sequence and of course the model:

$$\xi_t(i,j) = P(q_t = i, q_{t+1} = j|O,\lambda) \tag{6.27}$$

In order to compute ξ_t, we first compute a probability which is similar to ξ_t, but differs in including the probability of the observation:

(6.28) $$\text{not-quite-}\xi_t(i,j) = P(q_t = i, q_{t+1} = j, O|\lambda)$$

Fig. 6.13 shows the various probabilities that go into computing not-quite-ξ_t: the transition probability for the arc in question, the α probability before the arc, the β probability after the arc, and the observation probability for the symbol just after the arc.

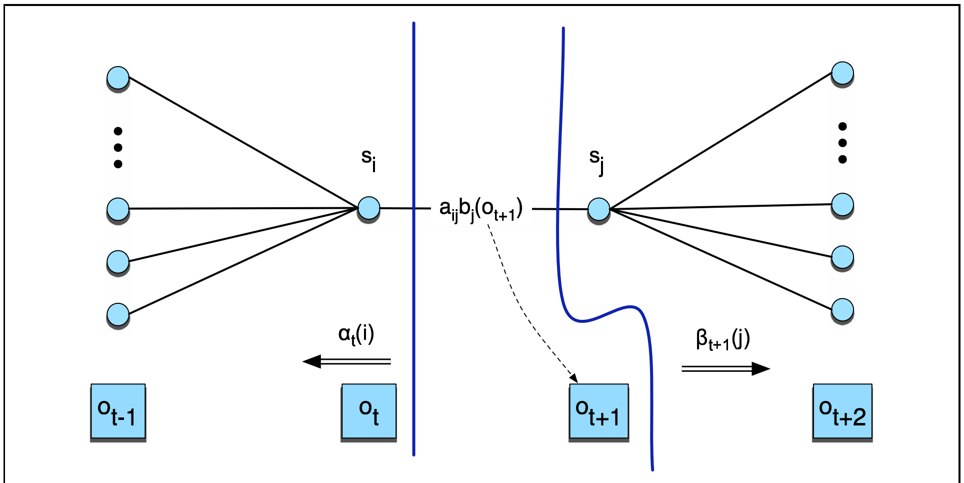


Figure 6.13 Computation of the joint probability of being in state i at time t and state j at time $t+1$. The figure shows the various probabilities that need to be combined to produce $P(q_t = i, q_{t+1} = j, O|\lambda)$: the α and β probabilities, the transition probability a_{ij} and the observation probability $b_j(o_{t+1})$. After Rabiner (1989).

These are multiplied together to produce *not-quite-*ξ_t as follows:

(6.29) $$\text{not-quite-}\xi_t(i,j) = \alpha_t(i)\, a_{ij} b_j(o_{t+1})\beta_{t+1}(j)$$

In order to compute ξ_t from *not-quite-*ξ_t, the laws of probability instruct us to divide by $P(O|\lambda)$, since:

(6.30) $$P(X|Y,Z) = \frac{P(X,Y|Z)}{P(Y|Z)}$$

The probability of the observation given the model is simply the forward probability of the whole utterance, (or alternatively the backward probability of the whole utterance!), which can thus be computed in a number of ways:

(6.31) $$P(O|\lambda) = \alpha_T(N) = \beta_T(1) = \sum_{j=1}^{N} \alpha_t(j)\beta_t(j)$$

So, the final equation for ξ_t is:

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij} b_j(o_{t+1})\beta_{t+1}(j)}{\alpha_T(N)} \tag{6.32}$$

The expected number of transitions from state i to state j is then the sum over all t of ξ. For our estimate of a_{ij} in (6.26), we just need one more thing: the total expected number of transitions from state i. We can get this by summing over all transitions out of state i. Here's the final formula for $\hat{a}_{ij}$:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} \xi_t(i,j)} \tag{6.33}$$

We also need a formula for recomputing the observation probability. This is the probability of a given symbol v_k from the observation vocabulary V, given a state j: $\hat{b}_j(v_k)$. We will do this by trying to compute:

$$\hat{b}_j(v_k) = \frac{\text{expected number of times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j} \tag{6.34}$$

For this we will need to know the probability of being in state j at time t, which we will call $\gamma_t(j)$:

$$\gamma_t(j) = P(q_t = j|O,\lambda) \tag{6.35}$$

Once again, we will compute this by including the observation sequence in the probability:

$$\gamma_t(j) = \frac{P(q_t = j, O|\lambda)}{P(O|\lambda)} \tag{6.36}$$

As Fig. 6.14 shows, the numerator of (6.36) is just the product of the forward probability and the backward probability:

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{P(O|\lambda)} \tag{6.37}$$

We are ready to compute b. For the numerator, we sum $\gamma_t(j)$ for all time steps t in which the observation o_t is the symbol v_k that we are interested in. For the denominator, we sum $\gamma_t(j)$ over all time steps t. The result will be the percentage of the times that we were in state j that we saw symbol v_k (the notation $\sum_{t=1 s.t. O_t=v_k}^{T}$ means ”sum over all t for which the observation at time t was v_k):

$$\hat{b}_j(v_k) = \frac{\sum_{t=1 s.t. O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{6.38}$$

We now have ways in (6.33) and (6.38) to *re-estimate* the transition A and observation B probabilities from an observation sequence O assuming that we already have a previous estimate of A and B.

These re-estimations form the core of the iterative forward-backward algorithm.

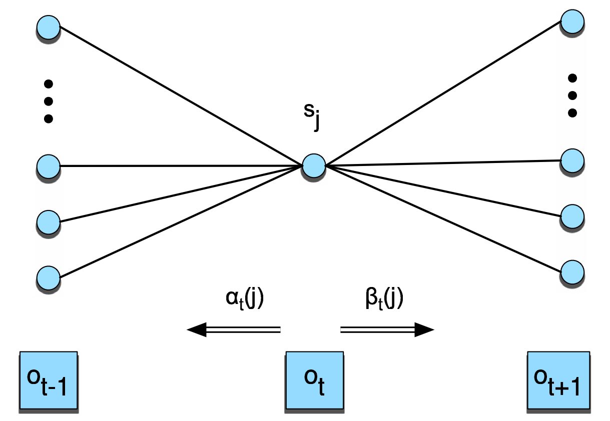


Figure 6.14 The computation of $\gamma_t(j)$, the probability of being in state j at time t. Note that ξ is really a degenerate case of ξ and hence this figure is like a version of Fig. 6.13 with state i collapsed with state j. After Rabiner (1989).

The forward-backward algorithm starts with some initial estimate of the HMM parameters $\lambda = (A, B, \pi)$. We then iteratively run two steps. Like other cases of the EM (expectation-maximization) algorithm the forward-backward algorithm has two steps: the **expectation** step, or **E-step**, and the **maximization** step, or **M-step**.

EXPECTATION
E-STEP
MAXIMIZATION
M-STEP

In the E-step we compute the expected state occupancy count γ and the expected state transition count ξ, from the earlier A and B probabilities. In the M-step, we use γ and ξ to recompute new A, B, and π probabilities.

Although in principle the Forward-Backward algorithm can do completely unsupervised learning of the A, B, and π parameters, in practice the initial conditions are very important. For this reason the algorithm is often given extra information. For example, for speech recognition, in practice the HMM structure is very often set by hand, and only the emission (B) and (non-zero) A transition probabilities are trained from a set of observation sequences O. Gaussian functions. Sec. **??** will also discuss how initial estimates for a and b are derived in speech recognition. We will also see in Ch. 9 that the forward-backward algorithm can be extended to inputs which are non-discrete (“continuous observation densities”).

6.6 LOGLINEAR MODELS

6.7 MAXIMUM ENTROPY MARKOV MODELS (MEMMS)

```
function FORWARD-BACKWARD(observations of len $T$, output vocabulary $V$, hidden
state set $Q$) returns HMM $A$, $B$, $\pi$

  initialize $A$, $B$, and $\pi$
  iterate until convergence
    E-step
      $\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{P(O|\lambda)} \; \forall\, t \text{ and } j$
      $\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij} b_j(o_{t+1})\beta_{t+1}(j)}{\alpha_T(N)} \; \forall\, t,\ i,\ \text{and } j$
    M-step
      $\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} \xi_t(i,j)}$
      $\hat{b}_j(v_k) = \frac{\sum_{t=1 s.t.\ O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$
  return $A$, $B$, $\pi$
```

Figure 6.15 The forward-backward algorithm.

6.8 EVALUATION AND STATISTICS

BIBLIOGRAPHICAL AND HISTORICAL NOTES

Baum, L. E. (1972). An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. In Shisha, O. (Ed.), *Inequalities III: Proceedings of the Third Symposium on Inequalities*, University of California, Los Angeles, pp. 1–8. Academic Press.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the *EM* algorithm. *Journal of the Royal Statistical Society*, *39*(1), 1–21.

Eisner, J. (2002). An interactive spreadsheet for teaching the forward-backward algorithm. In *Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching NLP and CL*, pp. 10–18.

Rabiner, L. R. (1989). A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257–286.