1. The forward message \( f_{t:t} = P(X_t \mid e_{1:t}) \) is replaced by the message
\[
\alpha_{i:t} = \max_{x_{t-1}} P(x_{t-1}, x_t \mid e_{1:t})
\]
that is, the probabilities of the most likely path to each state \( x_t \); and

2. the summation over \( x_t \) in Equation (15.5) is replaced by the maximization over \( x_t \) in Equation (15.11).

Thus, the algorithm for computing the most likely sequence is similar to filtering; it runs forward along the sequence, computing the message at each time step, using Equation (15.11). The progress of this computation is shown in Figure 15.5(b). At the end, it will have the probability for the most likely sequence overall (the states outlined in bold). In order to identify the actual sequence, as opposed to just computing its probability, the algorithm will also need to record, for each state, the best state that leads to it; these are indicated by the bold arrows in Figure 15.5(b). The optimal sequence is identified by following these bold arrows backwards from the best final state.

The algorithm we have just described is called the **Viterbi algorithm**, after its inventor. Unlike filtering, which uses constant space, its space requirement is also linear in \( t \). This is because the Viterbi algorithm needs to keep the pointers that identify the best sequence leading to each state.

### 15.3 Hidden Markov Models

The preceding section developed algorithms for temporal probabilistic reasoning using a general framework that was independent of the specific form of the transition and sensor models. In this and the next two sections, we discuss more concrete models and applications that illustrate the power of the basic algorithms and in some cases allow further improvements.

We begin with the hidden **Markov model**, or **HMM**. An HMM is a temporal probabilistic model in which the state of the process is described by a single discrete random variable. The possible values of the variable are the possible states of the world. The umbrella example described in the preceding section is therefore an HMM, since it has just one state variable. What happens if you have a model with two or more state variables? You can still fit it into the HMM framework by combining the variables into a single “megavariable” whose values are all possible tuples of values of the individual state variables. We will see that the restricted structure of HMMs allows for a simple and elegant matrix implementation of all the basic algorithms.  

---

The reader unfamiliar with basic operations on vectors and matrices might wish to consult Appendix A before proceeding with this section.
15.3.1 Simplified matrix algorithms

With a single, discrete state variable \( X_t \), we can give concrete form to the representations of the transition model, the sensor model, and the forward and backward messages. Let the state variable \( X_t \) have values denoted by integers \( 1, \ldots, S \), where \( S \) is the number of possible states. The transition model \( P(X_{t+1} | X_t) \) becomes an \( S \times S \) matrix \( T \), where

\[
T = P(X_{t+1} = j | X_t = i) .
\]

That is, \( T_{i,j} \) is the probability of a transition from state \( i \) to state \( j \). For example, the transition matrix for the umbrella world is

\[
T = \begin{pmatrix}
0.7 & 0.3 \\
0.3 & 0.7 
\end{pmatrix}
\]

We also put the sensor model in matrix form. In this case, because the value of the evidence variable \( e_t \) is known at time \( t \) (call it \( e_t \)), we need only specify, for each state, how likely it is that the state causes \( e_t \) to appear: we need \( P(e_t | X_t = i) \) for each state \( i \). For mathematical convenience we place these values into an \( S \times S \) diagonal matrix, \( O_t \), whose \( i \)th diagonal entry is \( P(e_t | X_t = i) \) and whose other entries are 0. For example, on day 1 in the umbrella world of Figure 15.5, \( U_1 = \text{true} \), and on day 3, \( U_3 = \text{false} \), so, from Figure 15.2, we have

\[
O_t = \begin{pmatrix}
0.9 & 0 & 0 \\
0 & 0.2 & 0 \\
0 & 0.1 & 0.8 
\end{pmatrix}
\]

Now, if we use column vectors to represent the forward and backward messages, all the computations become simple matrix-vector operations. The forward equation (15.5) becomes

\[
t = \mathbf{f}_t = \mathbf{O}_t \mathbf{f}_{t+1} T \mathbf{f}_{t+1} \quad \text{for } t = 1, \ldots, T .
\]

and the backward equation (15.9) becomes

\[
\mathbf{b}_{k+1} = \mathbf{O}_{k+1} \mathbf{b}_{k+2} .
\]

From these equations, we can see that the time complexity of the forward-backward algorithm (Figure 15.4) applied to a sequence of length \( t \) is \( O(S t) \), because each step requires multiplying an \( S \)-element vector by an \( S \times S \) matrix. The space requirement is \( O(S) \), because the forward pass stores \( t \) vectors of size \( S \).

Besides providing an elegant description of the filtering and smoothing algorithms for HMMs, the matrix formulation reveals opportunities for improved algorithms. The first is a simple variation on the forward-backward algorithm that allows smoothing to be carried out in constant space, independently of the length of the sequence. The idea is that smoothing for any particular time slice \( k \) requires the simultaneous presence of both the forward and backward messages, \( \mathbf{f}_{k:t} \) and \( \mathbf{b}_{k+1:t} \), according to Equation (15.8). The forward-backward algorithm achieves this by storing the \( f \) computed on the forward pass so that they are available during the backward pass. Another way to achieve this is with a single pass that propagates both \( \mathbf{f} \) and \( \mathbf{b} \) in the same direction. For example, the "forward" message \( \mathbf{f} \) can be propagated backward if we manipulate Equation (15.12) to work in the other direction:

\[
\mathbf{f}_t = \mathbf{O}_t \mathbf{f}_{t+1} T \mathbf{f}_{t+1} \quad \text{for } t = 1, \ldots, T .
\]

The modified smoothing algorithm works by first running the standard forward pass to compute \( \mathbf{f}_t \) (forgetting all the intermediate results) and then running the backward pass for both