國立成功大學
National Cheng Kung University
1931

Introduction to

Artificial Intelligence

Chapter 14
Probabilistic Reasoning over Time

Wei-Ta Chu (朱威達)

Time and Uncertainty

- We have mentioned probabilistic reasoning in the context of static worlds.
- Now consider a slightly different problem: treating a diabetic (糖尿病) patient. We have evidence such as recent insulin doses (胰島素劑量), food intake (攝取), blood sugar measurements, and other physical signs. The task is to assess the current state of the patient, including the actual blood sugar level and insulin level. Given this information, we can make a decision about the patient's food intake and insulin dose.

- Blood sugar levels and measurements thereof can change rapidly over time. We must model these changes.

States and observations

- We view the world as a series of snapshots, or **time slices**, each of which contains a set of random variables, some observable and some not.
- We assume some subset of variables is observable in each time slice. We will use $\mathbf{X}_t$ to denote the set of state variables at time t, which are assumed to be unobservable, and $\mathbf{E}_t$ to denote the set of observable evidence variables. The observation at time t is $\mathbf{E}_t = \mathbf{e}_t$ for some set of values $\mathbf{e}_t$.

States and observations

- You are the security guard stationed at a secret underground installation. You want to know whether it's raining today, but your only access to the outside world occurs each morning when you see the director coming in with, or without, an umbrella.

- For each day t, the set $\mathbf{E}_t$ thus contains a single evidence variable $Umbrella_t$ or U_t for short, and the set $\mathbf{X}_t$ contains a single state variable $Rain_t$ or R_t for short.

States and observations

- The interval between time slices also depends on the problem. For diabetes monitoring, a suitable interval might be an hour rather than a day. In this chapter we assume the interval between slices is fixed, so we can label times by integers.
- Our umbrella world is represented by state variables R_0, R_1, R_2, … and evidence variables U_1, U_2, ... We will use the notation $a{:}b$ to denote the sequence of integers from a to b (inclusive), and the notation $\mathbf{X}_{a:b}$ to denote the set of variables from $\mathbf{X}_a$ to $\mathbf{X}_b$.

Transition and sensor models

- The transition model specifies the probability distribution over the latest state variables, given the previous values, that is, $\mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{0:t-1})$.
- Now we face a problem: the set $\mathbf{X}_{0:t-1}$ is unbounded in size as t increases. We solve the problem by making a **Markov assumption**— that the current state depends on only a finite fixed number of previous states.
- The simplest case: the first-order Markov process, in which the current state depends only on the previous state and not on any earlier states.

Transition and sensor models

- With first-order Markov assumption, we have

$$\mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{0:t-1}) = \mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{t-1})$$

- Hence, in a first-order Markov process, the transition model is the conditional distribution $\mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{t-1})$. The transition model for a second-order Markov process is the conditional distribution $\mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{t-2}, \mathbf{X}_{t-1})$.
- Figure 15.1 shows the Bayesian network structures corresponding to first-order and second-order Markov processes.

Transition and sensor models

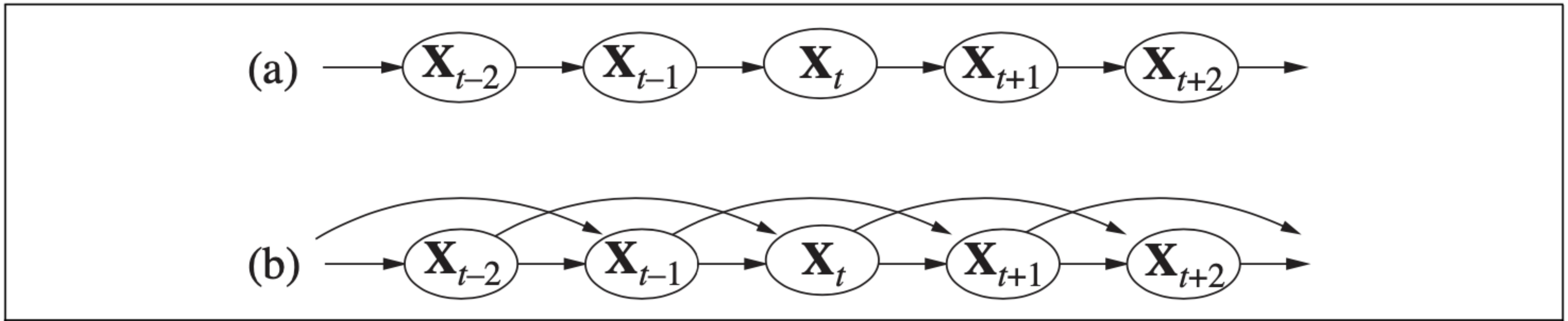


Figure 15.1 (a) Bayesian network structure corresponding to a first-order Markov process with state defined by the variables $\mathbf{X}_t$. (b) A second-order Markov process.

Transition and sensor models

- Even with the Markov assumption there is still a problem: there are infinitely many possible values of *t*. Do we need to specify a different distribution for each time step?
- We avoid this problem by assuming that changes in the world state are caused by a **stationary process**—that is, a process of change that is governed by laws that do not themselves change over time. (Don't confuse stationary with static: in a static process, the state itself does not change.)
- In the umbrella world, then, the conditional probability of rain, $\mathbf{P}(R_t \mid R_{t-1})$, is the same for all *t*, and we only have to specify one conditional probability table.

Transition and sensor models

- The sensor model: The evidence variables $\mathbf{E}_t$ could depend on previous variables as well as the current state variables. But, we make a sensor Markov assumption as follows:

$$\mathbf{P}(\mathbf{E}_t \mid \mathbf{X}_{0:t}, \mathbf{E}_{0:t-1}) = \mathbf{P}(\mathbf{E}_t \mid \mathbf{X}_t)$$

  Thus, $\mathbf{P}(\mathbf{E}_t \mid \mathbf{X}_t)$ is our sensor model (sometimes called the **observation model**).

- Figure 15.2 shows both the transition model and the sensor model for the umbrella example.

Transition and sensor models

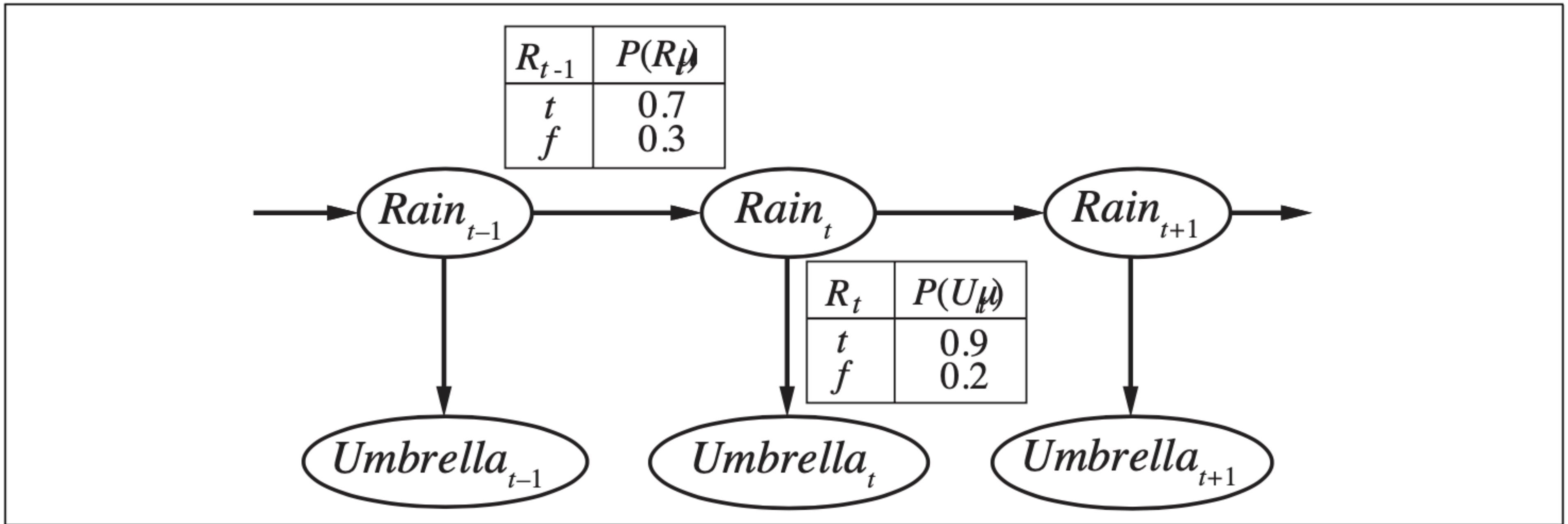


Figure 15.2 Bayesian network structure and conditional distributions describing the umbrella world. The transition model is $P(Rain_t \mid Rain_{t-1})$ and the sensor model is $P(Umbrella_t \mid Rain_t)$.

Transition and sensor models

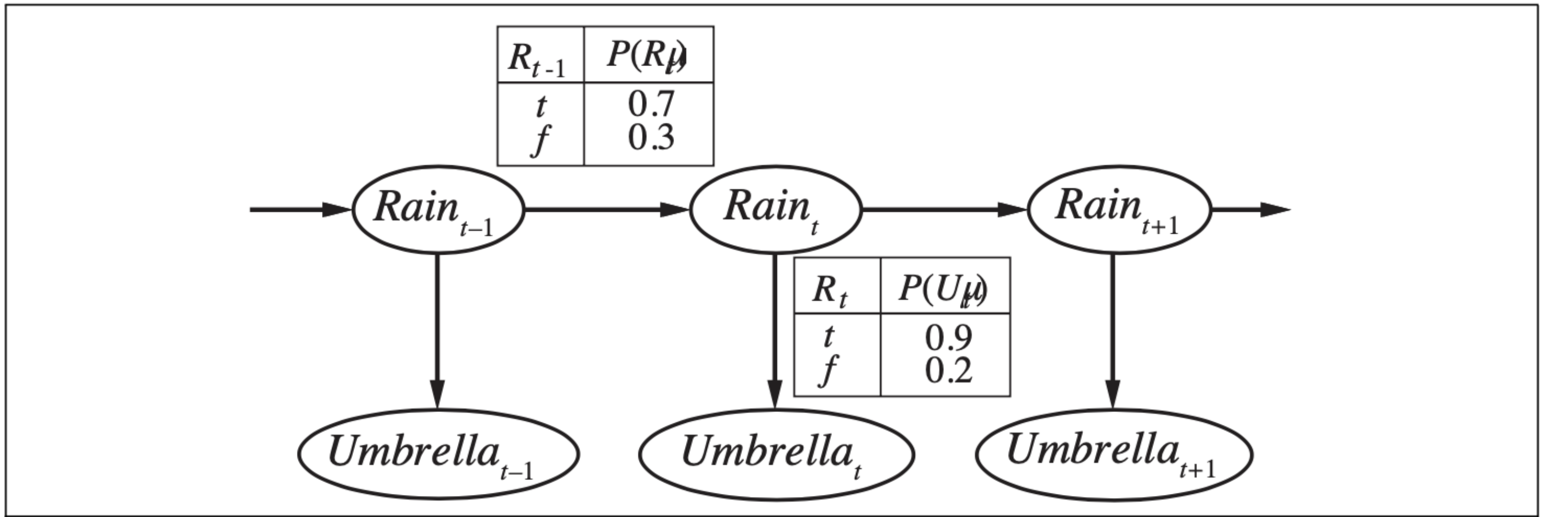
- Further, we need to say how everything gets started—the prior probability distribution at time 0, $\mathbf{P}(\mathbf{X}_0)$. With that, we have a specification of the complete joint distribution over all the variables, using Equation (14.2). For any t,

$$\mathbf{P}(\mathbf{X}_{0:t}, \mathbf{E}_{1:t}) = \mathbf{P}(\mathbf{X}_0) \prod_{i=1}^{t} \mathbf{P}(\mathbf{X}_i \mid \mathbf{X}_{i-1}) \mathbf{P}(\mathbf{E}_i \mid \mathbf{X}_i) \qquad (15.3)$$

The three terms on the right-hand side are the *initial state model* $\mathbf{P}(\mathbf{X}_0)$, the *transition model* $\mathbf{P}(\mathbf{X}_i \mid \mathbf{X}_{i-1})$, and the *sensor model* $\mathbf{P}(\mathbf{E}_i \mid \mathbf{X}_i)$.

Transition and sensor models

- First-order Markov process—the probability of rain is assumed to depend only on whether it rained the previous day. Whether such an assumption is reasonable depends on the domain itself. Sometimes the assumption is exactly true.



R_{t-1}	$P(R_t)$
t	0.7
f	0.3

R_t	$P(U_t)$
t	0.9
f	0.2

Figure 15.2 Bayesian network structure and conditional distributions describing the umbrella world. The transition model is $P(Rain_t \mid Rain_{t-1})$ and the sensor model is $P(Umbrella_t \mid Rain_t)$.

Transition and sensor models

- Sometimes the assumption is only approximate. There are two ways to improve the accuracy of the approximation:
    - Increasing the order of the Markov process model. For example, we could make a second-order model by adding $Rain_{t-2}$ as a parent of $Rain_t$.
    - Increasing the set of state variables. For example, we could add $Season_t$ to allow us to incorporate historical records of rainy seasons, or we could add $Temperature_t$, $Humidity_t$ and $Pressure_t$ to allow us to use a physical model of rainy conditions.

Inference in Temporal Models

- We can formulate the basic inference tasks that must be solved:
  - **Filtering**: This is the task of computing the **belief state**—the posterior distribution over the most recent state—given all evidence to date. Filtering is also called **state estimation**. In our example, we wish to compute $\mathbf{P}(\mathbf{X}_t \mid \mathbf{e}_{1:t})$. In the umbrella example, this would mean computing the probability of rain today, given all the observations of the umbrella carrier made so far.

Inference in Temporal Models

- We can formulate the basic inference tasks that must be solved:
  - **Prediction**: This is the task of computing the posterior distribution over the *future* state, given all evidence to date. That is, we wish to compute $\mathbf{P}(\mathbf{X}_{t+k} \mid \mathbf{e}_{1:t})$ for some $k > 0$. In the umbrella example, this might mean computing the probability of rain three days from now, given all the observations to date.

Inference in Temporal Models

- We can formulate the basic inference tasks that must be solved:
    - **Smoothing**: This is the task of computing the posterior distribution over a past state, given all evidence up to the present. That is, we wish to compute $\mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:t})$ for some k such that $0 \leq k < t$. In the umbrella example, it might mean computing the probability that it rained last Wednesday, given all the observations of the umbrella carrier made up to today. Smoothing provides a better estimate of the state than was available at the time, because it incorporates more evidence.

Inference in Temporal Models

- We can formulate the basic inference tasks that must be solved:
  - **Most likely explanation**: Given a sequence of observations, we might wish to find the sequence of states that is most likely to have generated those observations. That is, we wish to compute $\text{argmax}_{x1:t} P(\mathbf{x}_{1:t} \mid \mathbf{e}_{1:t})$. For example, if the umbrella appears on each of the first three days and is absent on the fourth, then the most likely explanation is that it rained on the first three days and did not rain on the fourth. Algorithms for this task are useful in many applications, including speech recognition—where the aim is to find the most likely sequence of words, given a series of sounds.

Inference in Temporal Models

- In addition to these inference tasks, we also have
  - **Learning**: The transition and sensor models can be learned from observations.

    The overall process is an instance of the expectation-maximization or **EM algorithm**.

Filtering and prediction

- A useful filtering algorithm needs to maintain a current state estimate and update it, rather than going back over the entire history of percepts for each update.
- In other words, given the result of filtering up to time *t*, the agent needs to compute the result for $t + 1$ from the new evidence $\mathbf{e}_{t+1}$,

$$\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t+1}) = f(\mathbf{e}_{t+1}, \mathbf{P}(\mathbf{X}_t \mid \mathbf{e}_{1:t}))$$

for some function *f*.

Filtering and prediction

- This process is called **recursive estimation**. We can view the calculation as being composed of two parts: first, the current state distribution is projected forward from *t* to *t*+1; then it is updated using the new evidence $\mathbf{e}_{t+1}$. This two-part process emerges quite simply when the formula is rearranged:

$$\begin{aligned}\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t+1}) &= \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t}, \mathbf{e}_{t+1}) \quad \text{(dividing up the evidence)}\\ &= \alpha\, \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}, \mathbf{e}_{1:t})\, \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t}) \quad \text{(using Bayes' rule)}\\ &= \alpha\, \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1})\, \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t}) \quad \text{(by the sensor Markov assumption).} \qquad (15.4)\end{aligned}$$

Filtering and prediction

- α is a normalizing constant used to make probabilities sum up to 1. The second term, $\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t})$ represents a one-step prediction of the next state, and $\mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1})$ is obtainable directly from the sensor model. Now we obtain the one-step prediction for the next state by conditioning on the current state $\mathbf{X}_t$:

$$\begin{aligned}\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t+1}) &= \alpha\,\mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}) \sum_{\mathbf{x}_t} \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t, \mathbf{e}_{1:t}) P(\mathbf{x}_t \mid \mathbf{e}_{1:t}) \\ &= \alpha\,\mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}) \sum_{\mathbf{x}_t} \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t) P(\mathbf{x}_t \mid \mathbf{e}_{1:t}) \quad \text{(Markov assumption).} \qquad (15.5)\end{aligned}$$

Filtering and prediction

- Within the summation, the first factor comes from the *transition model* and the second comes from *the current state distribution*. Hence, we have the desired recursive formulation. We can think of the filtered estimate $\mathbf{P}(\mathbf{X}_t \mid \mathbf{e}_{1:t})$ as a "message" $\mathbf{f}_{1:t}$ that is propagated forward along the sequence, modified by each transition and updated by each new observation. The process is given by

$$\mathbf{f}_{1:t+1} = \alpha\,\textsc{Forward}(\mathbf{f}_{1:t}, \mathbf{e}_{t+1})$$

  where FORWARD implements the update described in Equation (15.5) and the process begins with $\mathbf{f}_{1:0} = \mathbf{P}(\mathbf{X}_0)$.

$$\begin{aligned}\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t+1}) &= \alpha\,\mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}) \sum_{\mathbf{x}_t} \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t, \mathbf{e}_{1:t}) P(\mathbf{x}_t \mid \mathbf{e}_{1:t}) \\ &= \alpha\,\mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}) \sum_{\mathbf{x}_t} \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t) P(\mathbf{x}_t \mid \mathbf{e}_{1:t}) \quad \text{(Markov assumption).} \qquad (15.5)\end{aligned}$$

Filtering and prediction

Let us illustrate the filtering process for two steps in the basic umbrella example (Figure 15.2.) That is, we will compute $\mathbf{P}(R_2 \mid u_{1:2})$ as follows:

- On day 0, we have no observations, only the security guard's prior beliefs; let's assume that consists of $\mathbf{P}(R_0) = \langle 0.5, 0.5 \rangle$.
- On day 1, the umbrella appears, so $U_1 = \mathit{true}$. The prediction from $t=0$ to $t=1$ is

$$\begin{aligned}\mathbf{P}(R_1) &= \sum_{r_0} \mathbf{P}(R_1 \mid r_0) P(r_0) \\ &= \langle 0.7, 0.3 \rangle \times 0.5 + \langle 0.3, 0.7 \rangle \times 0.5 = \langle 0.5, 0.5 \rangle \ .\end{aligned}$$

Then the update step simply multiplies by the probability of the evidence for $t=1$ and normalizes, as shown in Equation (15.4):

$$\begin{aligned}\mathbf{P}(R_1 \mid u_1) &= \alpha \, \mathbf{P}(u_1 \mid R_1) \mathbf{P}(R_1) = \alpha \, \langle 0.9, 0.2 \rangle \langle 0.5, 0.5 \rangle \\ &= \alpha \, \langle 0.45, 0.1 \rangle \approx \langle 0.818, 0.182 \rangle \ .\end{aligned}$$

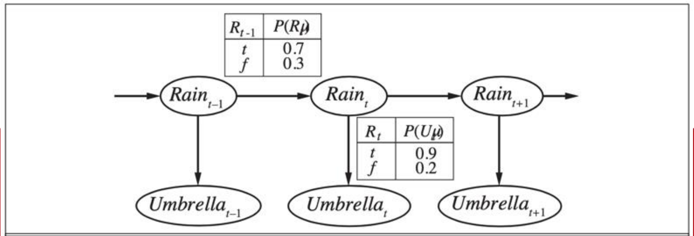


Figure 15.2 Bayesian network structure and conditional distributions describing the umbrella world. The transition model is $P(Rain_t \mid Rain_{t-1})$ and the sensor model is $P(Umbrella_t \mid Rain_t)$.

Filtering and prediction

- On day 2, the umbrella appears, so $U_2 = true$. The prediction from $t = 1$ to $t = 2$ is

$$\begin{aligned}\mathbf{P}(R_2 \mid u_1) &= \sum_{r_1} \mathbf{P}(R_2 \mid r_1) P(r_1 \mid u_1) \\ &= \langle 0.7, 0.3 \rangle \times 0.818 + \langle 0.3, 0.7 \rangle \times 0.182 \approx \langle 0.627, 0.373 \rangle \ ,\end{aligned}$$

  and updating it with the evidence for $t = 2$ gives

$$\begin{aligned}\mathbf{P}(R_2 \mid u_1, u_2) &= \alpha \, \mathbf{P}(u_2 \mid R_2) \mathbf{P}(R_2 \mid u_1) = \alpha \, \langle 0.9, 0.2 \rangle \langle 0.627, 0.373 \rangle \\ &= \alpha \, \langle 0.565, 0.075 \rangle \approx \langle 0.883, 0.117 \rangle \ .\end{aligned}$$

Intuitively, the probability of rain increases from day 1 to day 2 because rain persists.

Filtering and prediction

- The task of prediction can be seen simply as filtering without the addition of new evidence. In fact, the filtering process already incorporates a one-step prediction, and it is easy to derive the following recursive computation for predicting the state at $t + k + 1$ from a prediction for $t + k$:

$$\mathbf{P}(\mathbf{X}_{t+k+1} \mid \mathbf{e}_{1:t}) = \sum_{\mathbf{x}_{t+k}} \mathbf{P}(\mathbf{X}_{t+k+1} \mid \mathbf{x}_{t+k}) P(\mathbf{x}_{t+k} \mid \mathbf{e}_{1:t})$$

  Naturally, this computation involves only the transition model and not the sensor model.

Smoothing

- Smoothing is the process of computing the distribution over past states given evidence up to the present; that is, $\mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:t})$ for $0 \leq k < t$. (See Figure 15.3.) In anticipation of another recursive message-passing approach, we can split the computation into two parts—the evidence up to k and the evidence from $k + 1$ to t,

$$\begin{aligned}
\mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:t}) &= \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k}, \mathbf{e}_{k+1:t}) \\
&= \alpha\, \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k}) \mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k, \mathbf{e}_{1:k}) \quad \text{(using Bayes' rule)} \\
&= \alpha\, \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k}) \mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k) \quad \text{(using conditional independence)} \\
&= \alpha\, \mathbf{f}_{1:k} \times \mathbf{b}_{k+1:t}\,. \qquad (15.8)
\end{aligned}$$

where "×" represents pointwise multiplication of vectors.

Smoothing

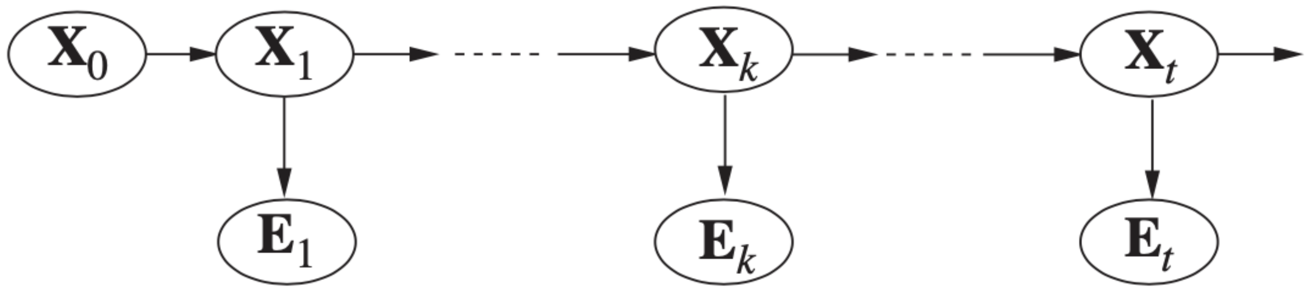


Figure 15.3 Smoothing computes $\mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:t})$, the posterior distribution of the state at some past time k given a complete sequence of observations from 1 to t.

Smoothing

- Here we have defined a "backward" message $\mathbf{b}_{k+1:t} = \mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k)$, analogous to the forward message $\mathbf{f}_{1:k}$. The forward message $\mathbf{f}_{1:k}$ can be computed by filtering forward from 1 to k. It turns out that the backward message $\mathbf{b}_{k+1:t}$ can be computed by a recursive process that runs backward from t:

$$
\begin{aligned}
\mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k) &= \sum_{\mathbf{x}_{k+1}} \mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k, \mathbf{x}_{k+1}) \mathbf{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k) \quad \text{(conditioning on } \mathbf{X}_{k+1}\text{)} \\
&= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1:t} \mid \mathbf{x}_{k+1}) \mathbf{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k) \quad \text{(by conditional independence)} \\
&= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1}, \mathbf{e}_{k+2:t} \mid \mathbf{x}_{k+1}) \mathbf{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k) \\
&= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1} \mid \mathbf{x}_{k+1}) P(\mathbf{e}_{k+2:t} \mid \mathbf{x}_{k+1}) \mathbf{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k) , \qquad (15.9)
\end{aligned}
$$

Smoothing

- The last step follows by the conditional independence of $\mathbf{e}_{k+1}$ and $\mathbf{e}_{k+2:t}$, given $\mathbf{X}_{k+1}$. Of the three factors in this summation, the first and third are obtained directly from the model, and the second is the "recursive call." Using the message notation, we have

$$\mathbf{b}_{k+1:t} = \text{BACKWARD}(\mathbf{b}_{k+2:t}, \mathbf{e}_{k+1})$$

  where BACKWARD implements the update described in Equation (15.9).

Smoothing

- We can now see that the two terms in Equation (15.8) can both be computed by recursions through time, one running forward from 1 to k and using the filtering equation (15.5) and the other running backward from t to $k + 1$ and using Equation (15.9).
- The backward phase is initialized with $\mathbf{b}_{t+1:t} = \mathbf{P}(\mathbf{e}_{t+1:t} \mid \mathbf{X}_t) = \mathbf{P}(\ \mid \mathbf{X}_t)\mathbf{1}$, where $\mathbf{1}$ is a vector of 1s. (Because $\mathbf{e}_{t+1:t}$ is an empty sequence, the probability of observing it is 1.)

$$\begin{aligned}
\mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:t}) &= \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k}, \mathbf{e}_{k+1:t}) \\
&= \alpha\, \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k})\mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k, \mathbf{e}_{1:k}) \quad \text{(using Bayes' rule)} \\
&= \alpha\, \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k})\mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k) \quad \text{(using conditional independence)} \\
&= \alpha\, \mathbf{f}_{1:k} \times \mathbf{b}_{k+1:t}\,. \qquad (15.8)
\end{aligned}$$

Smoothing

Let us now apply this algorithm to the umbrella example, computing the smoothed estimate for the probability of rain at time $k=1$, given the umbrella observations on days 1 and 2. From Equation (15.8), this is given by

$$\mathbf{P}(R_1 \mid u_1, u_2) = \alpha\, \mathbf{P}(R_1 \mid u_1)\, \mathbf{P}(u_2 \mid R_1)\ . \tag{15.10}$$

The first term we already know to be $\langle .818, .182 \rangle$, from the forward filtering process described earlier. The second term can be computed by applying the backward recursion in Equation (15.9):

$$\begin{aligned}\mathbf{P}(u_2 \mid R_1) &= \sum_{r_2} P(u_2 \mid r_2) P(\ \mid r_2) \mathbf{P}(r_2 \mid R_1)\\ &= (0.9 \times 1 \times \langle 0.7, 0.3\rangle) + (0.2 \times 1 \times \langle 0.3, 0.7\rangle) = \langle 0.69, 0.41\rangle\ .\end{aligned}$$

Plugging this into Equation (15.10), we find that the smoothed estimate for rain on day 1 is

$$\mathbf{P}(R_1 \mid u_1, u_2) = \alpha\, \langle 0.818, 0.182\rangle \times \langle 0.69, 0.41\rangle \approx \langle 0.883, 0.117\rangle\ .$$

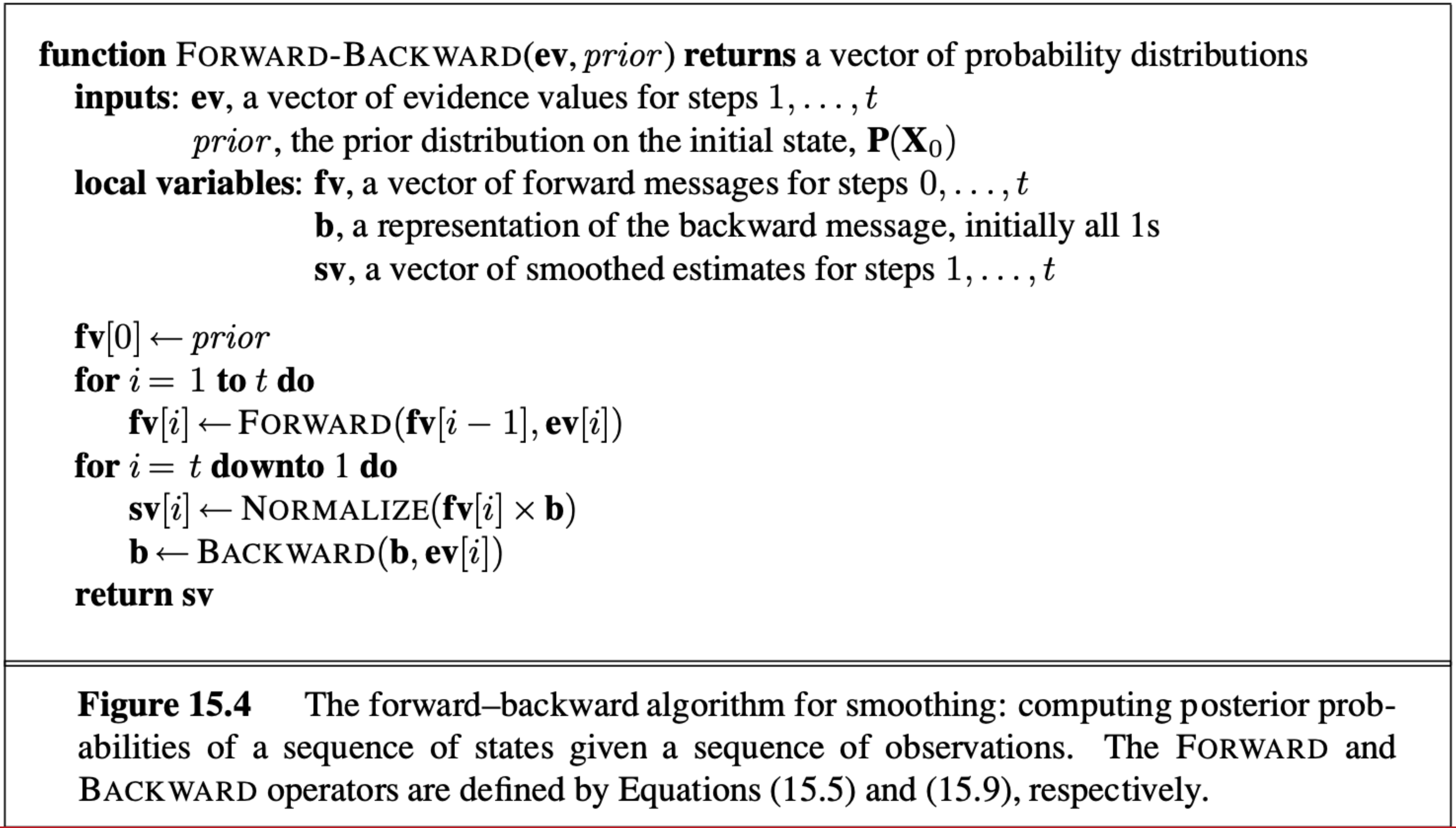
Smoothing

- Thus, the smoothed estimate for rain on day 1 is higher than the filtered estimate (0.818) in this case. This is because the umbrella on day 2 makes it more likely to have rained on day 2; in turn, because rain tends to persist, that makes it more likely to have rained on day 1.
- Both the forward and backward recursions take a constant amount of time per step; hence, the time complexity of smoothing with respect to evidence $\mathbf{e}_{1:t}$ is $O(t)$. This is the complexity for smoothing at a particular time step k. If we want to smooth the whole sequence, one obvious method is simply to run the whole smoothing process once for each time step to be smoothed. This results in a time complexity of $O(t^2)$.

Smoothing

- A better approach uses dynamic programming to reduce the complexity to $O(t)$.
- The key to the linear-time algorithm is to record the results of forward filtering over the whole sequence. Then we run the backward recursion from t down to 1, computing the smoothed estimate at each step k from the computed backward message $\mathbf{b}_{k+1:t}$ and the stored forward message $\mathbf{f}_{1:k}$. The algorithm alled the **forward–backward algorithm** is shown in Figure 15.4.

Smoothing



```
function FORWARD-BACKWARD(ev, prior) returns a vector of probability distributions
  inputs: ev, a vector of evidence values for steps $1, \ldots, t$
          prior, the prior distribution on the initial state, $\mathbf{P}(\mathbf{X}_0)$
  local variables: fv, a vector of forward messages for steps $0, \ldots, t$
                   b, a representation of the backward message, initially all 1s
                   sv, a vector of smoothed estimates for steps $1, \ldots, t$

  $\mathbf{fv}[0] \leftarrow prior$
  for $i = 1$ to $t$ do
      $\mathbf{fv}[i] \leftarrow$ FORWARD($\mathbf{fv}[i-1], \mathbf{ev}[i]$)
  for $i = t$ downto $1$ do
      $\mathbf{sv}[i] \leftarrow$ NORMALIZE($\mathbf{fv}[i] \times \mathbf{b}$)
      $\mathbf{b} \leftarrow$ BACKWARD($\mathbf{b}, \mathbf{ev}[i]$)
  return sv
```

Figure 15.4 The forward–backward algorithm for smoothing: computing posterior probabilities of a sequence of states given a sequence of observations. The FORWARD and BACKWARD operators are defined by Equations (15.5) and (15.9), respectively.

Finding the most likely sequence

- Suppose that [*true*, *true*, *false*, *true*, *true*] is the umbrella sequence for the security guard's first five days on the job.
- In all, there are 2^5 possible weather sequences we could pick. Is there a way to find the most likely one, short of enumerating all of them?

- We could try this linear-time procedure: use smoothing to find the posterior distribution for the weather at each time step; then construct the sequence, using at each step the weather that is most likely according to the posterior.

Finding the most likely sequence

- View each sequence as a path through a graph whose nodes are the possible states at each time step. Such a graph is shown for the umbrella world in Figure 15.5(a).
- Consider the task of finding the most likely path through this graph, where the likelihood of any path is the product of the transition probabilities along the path and the probabilities of the given observations at each state.

Finding the most likely sequence

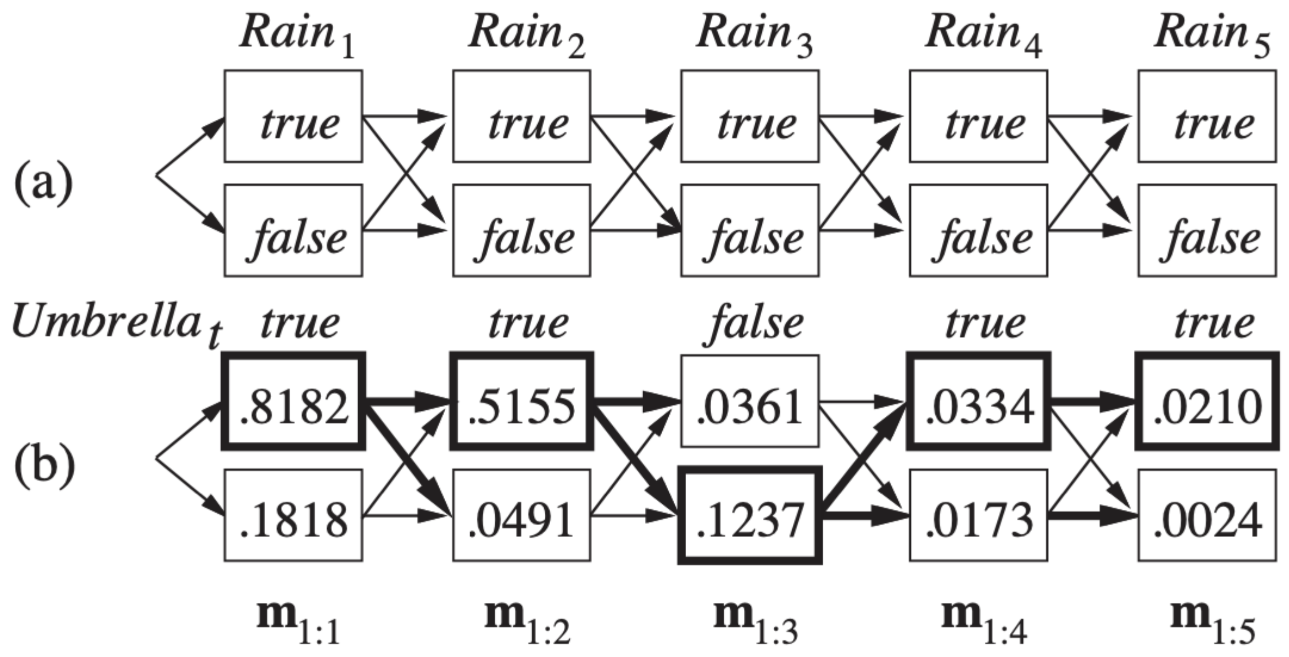


Figure 15.5 (a) Possible state sequences for $Rain_t$ can be viewed as paths through a graph of the possible states at each time step. (States are shown as rectangles to avoid confusion with nodes in a Bayes net.) (b) Operation of the Viterbi algorithm for the umbrella observation sequence [*true*, *true*, *false*, *true*, *true*]. For each t, we have shown the values of the message $\mathbf{m}_{1:t}$, which gives the probability of the best sequence reaching each state at time t. Also, for each state, the bold arrow leading into it indicates its best predecessor as measured by the product of the preceding sequence probability and the transition probability. Following the bold arrows back from the most likely state in $\mathbf{m}_{1:5}$ gives the most likely sequence.

Finding the most likely sequence

- Let's focus on paths that reach the state *Rain*$_5$ =*true*. Because of the Markov property, the most likely path to the state *Rain*$_5$ =*true* consists of the most likely path to *some* state at time 4 followed by a transition to *Rain*$_5$ =*true*; and the state at time 4 that will become part of the path to *Rain*$_5$ =*true* is whichever maximizes the likelihood of that path. In other words, there is a recursive relationship between most likely paths to each state $\mathbf{x}_{t+1}$ and most likely paths to each state $\mathbf{x}_t$.

$$\max_{\mathbf{x}_1 \ldots \mathbf{x}_t} \mathbf{P}(\mathbf{x}_1, \ldots, \mathbf{x}_t, \mathbf{X}_{t+1} \mid \mathbf{e}_{1:t+1})$$

$$= \alpha \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}) \max_{\mathbf{x}_t} \left(\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t) \max_{\mathbf{x}_1 \ldots \mathbf{x}_{t-1}} P(\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}, \mathbf{x}_t \mid \mathbf{e}_{1:t}) \right) . \quad (15.11)$$

$$\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t+1}) = \alpha \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}) \sum_{\mathbf{x}_t} \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t, \mathbf{e}_{1:t}) P(\mathbf{x}_t \mid \mathbf{e}_{1:t})$$

$$= \alpha \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}) \sum_{\mathbf{x}_t} \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t) P(\mathbf{x}_t \mid \mathbf{e}_{1:t}) \quad \text{(Markov assumption).} \qquad (15.5)$$

Finding the most likely sequence

- Equation (15.11) is identical to the filtering equation (15.5) except that

1. The forward message $\mathbf{f}_{1:t} = \mathbf{P}(\mathbf{X}_t \mid \mathbf{e}_{1:t})$ is replaced by the message

$$\mathbf{m}_{1:t} = \max_{\mathbf{x}_1 \ldots \mathbf{x}_{t-1}} \mathbf{P}(\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}, \mathbf{X}_t \mid \mathbf{e}_{1:t}) ,$$

   that is, the probabilities of the most likely path to each state $\mathbf{x}_t$; and

2. the summation over $\mathbf{x}_t$ in Equation (15.5) is replaced by the maximization over $\mathbf{x}_t$ in Equation (15.11).

Finding the most likely sequence

- Thus, the algorithm for computing the most likely sequence is similar to filtering: it runs forward along the sequence, computing the **m** 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 reaching *each* of the final states. One can thus easily select the most likely sequence overall (the states outlined in bold).
- The optimal sequence is identified by following these bold arrows backwards from the best final state.

Finding the most likely sequence

- The algorithm we have just described is called the **Viterbi algorithm**. Like the filtering algorithm, its time complexity is linear in *t*, the length of the sequence. 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.

Hidden Markov Models

- 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: $Rain_t$. 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.

HMM example: Localization

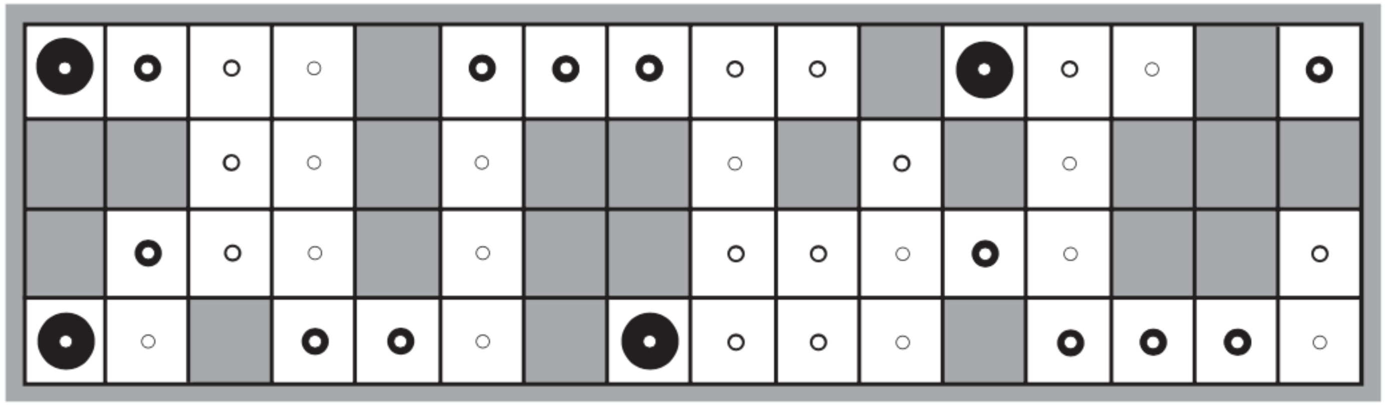
- Here we make the vacuum problem slightly more realistic by including a simple probability model for the robot's motion and by allowing for noise in the sensors. The state variable X_t represents the location of the robot on the discrete grid; the domain of this variable is the set of empty squares $\{s_1, \ldots, s_n\}$. Let NEIGHBORS(s) be the set of empty squares that are adjacent to *s* and let *N*(*s*) be the size of that set. Then the transition model for *Move* action says that the robot is equally likely to end up at any neighboring square:

$$P(X_{t+1} = j \mid X_t = i) = \mathbf{T}_{ij} = (1/N(i) \text{ if } j \in \textsc{Neighbors}(i) \text{ else } 0)$$

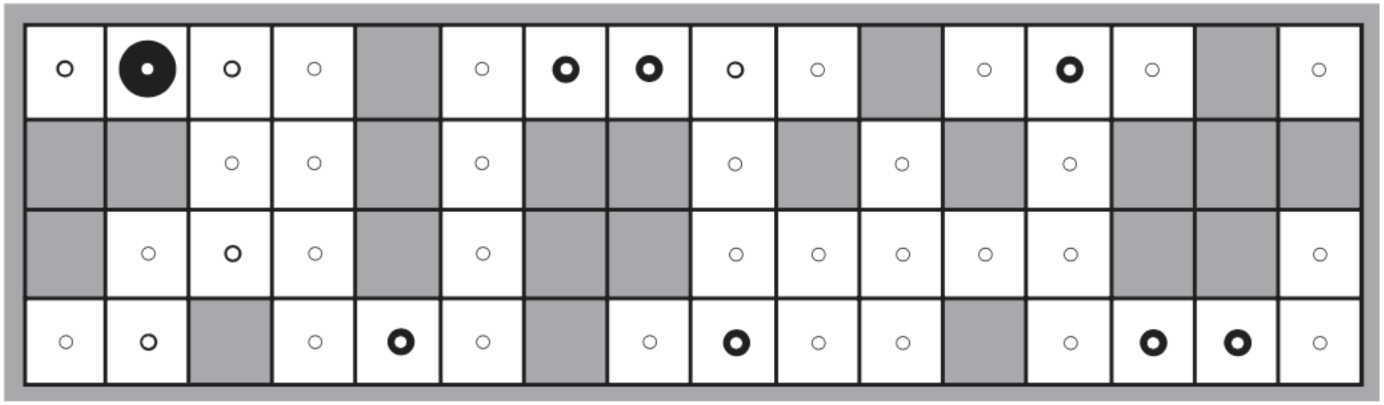
HMM example: Localization

- We don’t know where the robot starts, so we will assume a uniform distribution over all the squares; that is, $P(X_0 = i) = 1/n$. For the particular environment we consider (Figure 15.7), $n = 42$ and the transition matrix $\mathbf{T}$ has $42 \times 42 = 1764$ entries.

HMM example: Localization



(a) Posterior distribution over robot location after $E_1 = NSW$



(b) Posterior distribution over robot location after $E_1 = NSW, E_2 = NS$

Figure 15.7 Posterior distribution over robot location: (a) one observation $E_1 = NSW$; (b) after a second observation $E_2 = NS$. The size of each disk corresponds to the probability that the robot is at that location. The sensor error rate is $\epsilon = 0.2$.

HMM example: Localization

- The sensor variable E_t has 16 possible values, each a four-bit sequence giving the presence or absence of an obstacle in a particular compass direction. We will use the notation *NS*, for example, to mean that the north and south sensors report an obstacle and the east and west do not. Suppose that each sensor's error rate is ε and that errors occur independently for the four sensor directions. In that case, the probability of getting all four bits right is $(1-\varepsilon)^4$ and the probability of getting them all wrong is ε^4 . Furthermore, if d_{it} is the discrepancy—the number of bits that are different—between the true values for square i and the actual reading e_t, then the probability that a robot in square i would receive a sensor reading e_t is

$$P(E_t = e_t \mid X_t = i) = \mathbf{O}_{t_{ii}} = (1-\epsilon)^{4-d_{it}} \epsilon^{d_{it}}$$

HMM example: Localization

- For example, the probability that a square with obstacles to the north and south would produce a sensor reading *NSE* is $(1-\varepsilon)^3\varepsilon^1$.
- Given the matrices **T** and $\mathbf{O}_t$, the robot can compute the posterior distribution over locations to work out where it is.
- Figure 15.7 shows the distributions $\mathbf{P}(X_1 \mid E_1 = NSW)$ and $\mathbf{P}(X_2 \mid E_1 = NSW, E_2 = NS)$.

HMM example: Localization

- In addition to filtering to estimate its current location, the robot can use smoothing to work out where it was at any given past time and it can use the Viterbi algorithm to work out the most likely path it has taken to get where it is now.
- Figure 15.8 shows the localization error and Viterbi path accuracy for various values of the per-bit sensor error rate ε. Even when ε is 20%—which means that the overall sensor reading is wrong 59% of the time—the robot is usually able to work out its location within two squares after 25 observations.

HMM example: Localization

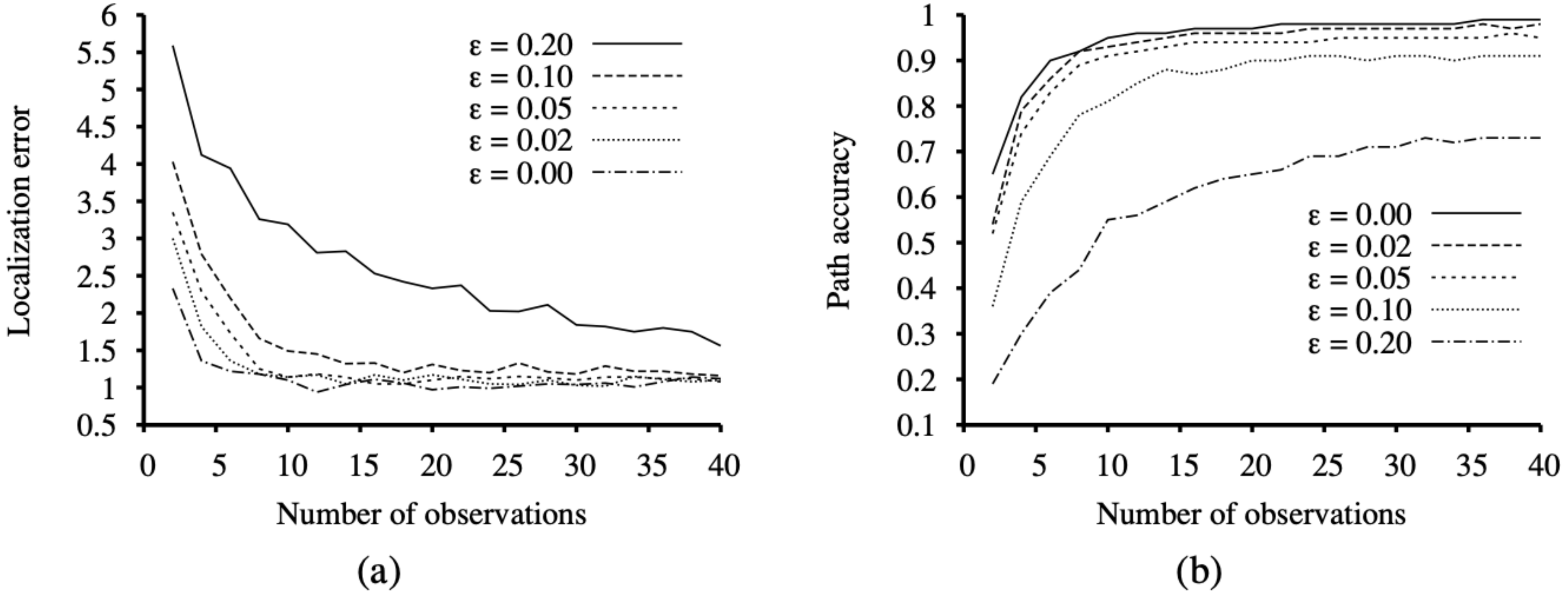


Figure 15.8 Performance of HMM localization as a function of the length of the observation sequence for various different values of the sensor error probability ϵ; data averaged over 400 runs. (a) The localization error, defined as the Manhattan distance from the true location. (b) The Viterbi path accuracy, defined as the fraction of correct states on the Viterbi path.

HMM example: Localization

- This is because of the algorithm’s ability to integrate evidence over time and to take into account the probabilistic constraints imposed on the location sequence by the transition model. When ε is 10%, the performance after a half-dozen observations is hard to distinguish from the performance with perfect sensing.
- Broadly speaking, high levels of localization and path accuracy are maintained even in the face of substantial errors in the models used.

Kalman Filters

- Imagine watching a small bird flying through dense jungle foliage at dusk: you glimpse brief, intermittent flashes of motion; you try hard to guess where the bird is and where it will appear next so that you don’t lose it. Or imagine that you are a World War II radar operator peering at a faint, wandering blip that appears once every 10 seconds on the screen.
- In all these cases, you are doing filtering: estimating state variables (here, position and velocity) from noisy observations over time. If the variables were discrete, we could model the system with a hidden Markov model. This section examines methods for handling continuous variables, using an algorithm called **Kalman filtering**.

Kalman Filters

- The bird's flight might be specified by six continuous variables at each time point; three for position (X_t, Y_t, Z_t) and three for velocity ($\underline{X}_t$, $\underline{Y}_t$, $\underline{Z}_t$). We will need suitable conditional densities to represent the transition and sensor models: linear Gaussian distributions. The next state X_{t+1} must be a linear function of the current state X_t, plus some Gaussian noise. Consider the X-coordinate of the bird. Let the time interval between observations be Δ, and assume constant velocity during the interval; then the position update is given by $X_{t+\Delta} = X_t + \underline{X}_t\Delta$. Adding Gaussian noise, we obtain a linear Gaussian transition model:

$$P(X_{t+\Delta} = x_{t+\Delta} \mid X_t = x_t, \dot{X}_t = \dot{x}_t) = N(x_t + \dot{x}_t\,\Delta, \sigma^2)(x_{t+\Delta})$$

Kalman Filters

- The Bayesian network structure for a system with position vector X_t and velocity $\underline{X}_t$ is shown in Figure 15.9.
- A multivariate Gaussian distribution for *d* variables is specified by a *d*-element mean $\boldsymbol{\mu}$ and a $d \times d$ covariance matrix Σ.

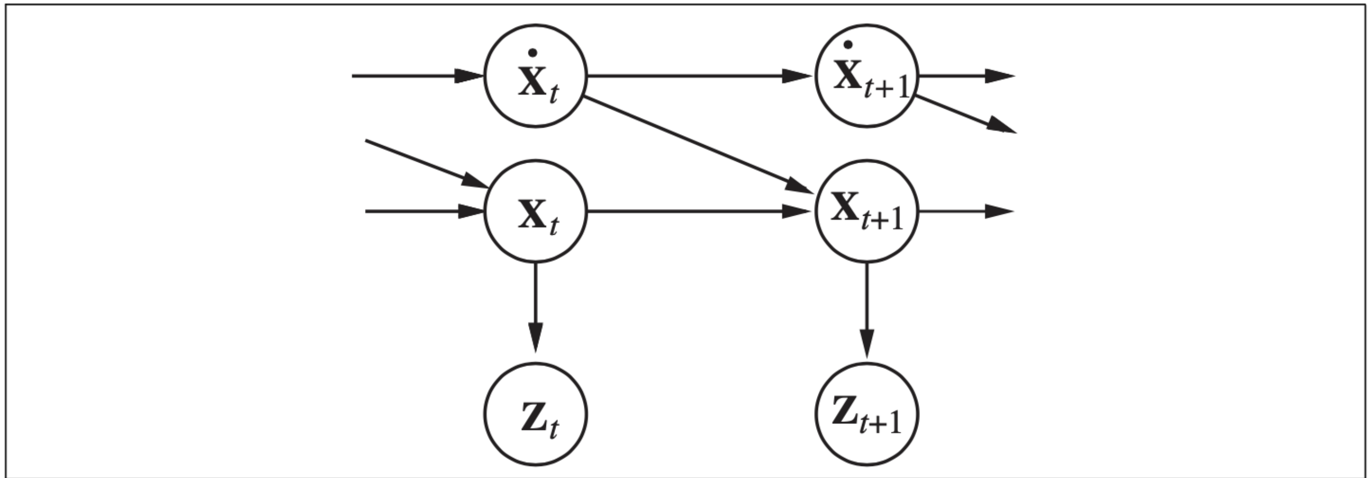


Figure 15.9 Bayesian network structure for a linear dynamical system with position $\mathbf{X}_t$, velocity $\dot{\mathbf{X}}_t$, and position measurement $\mathbf{Z}_t$.

Updating Gaussian distributions

- A key property of the linear Gaussian family of distributions: it remains closed under the standard Bayesian network operations.

1. If the current distribution $\mathbf{P}(\mathbf{X}_t \mid \mathbf{e}_{1:t})$ is Gaussian and the transition model $\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t)$ is linear Gaussian, then the one-step predicted distribution given by

$$\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t}) = \int_{\mathbf{x}_t} \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t) P(\mathbf{x}_t \mid \mathbf{e}_{1:t})\, d\mathbf{x}_t \tag{15.17}$$

is also a Gaussian distribution.

2. If the prediction $\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t})$ is Gaussian and the sensor model $\mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1})$ is linear Gaussian, then, after conditioning on the new evidence, the updated distribution

$$\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t+1}) = \alpha\, \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}) \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t}) \tag{15.18}$$

is also a Gaussian distribution.

Updating Gaussian distributions

- Thus, the FORWARD operator for Kalman filtering takes a Gaussian forward message $\mathbf{f}_{1:t}$, specified by a mean $\boldsymbol{\mu}_t$ and covariance matrix $\boldsymbol{\Sigma}_t$, and produces a new multivariate Gaussian forward message $\mathbf{f}_{1:t+1}$, specified by a mean $\boldsymbol{\mu}_{t+1}$ and covariance matrix $\boldsymbol{\Sigma}_{t+1}$.
- So, if we start with a Gaussian prior $\mathbf{f}_{1:0} = \mathbf{P}(\mathbf{X}_0) = N(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$, filtering with a linear Gaussian model produces a Gaussian state distribution for all time.

A simple one-dimensional example

- The temporal model we consider describes a random walk of a single continuous state variable X_t with a noisy observation Z_t.
- An example might be the "consumer confidence" index, which can be modeled as undergoing a random Gaussian-distributed change each month and is measured by a random consumer survey that also introduces Gaussian sampling noise.

A simple one-dimensional example

The prior distribution is assumed to be Gaussian with variance σ_0^2:

$$P(x_0) = \alpha\, e^{-\frac{1}{2}\left(\frac{(x_0-\mu_0)^2}{\sigma_0^2}\right)} .$$

(For simplicity, we use the same symbol α for all normalizing constants in this section.) The transition model adds a Gaussian perturbation of constant variance σ_x^2 to the current state:

$$P(x_{t+1} \mid x_t) = \alpha\, e^{-\frac{1}{2}\left(\frac{(x_{t+1}-x_t)^2}{\sigma_x^2}\right)} .$$

The sensor model assumes Gaussian noise with variance σ_z^2:

$$P(z_t \mid x_t) = \alpha\, e^{-\frac{1}{2}\left(\frac{(z_t-x_t)^2}{\sigma_z^2}\right)} .$$

Now, given the prior $\mathbf{P}(X_0)$, the one-step predicted distribution comes from Equation (15.17):

$$\begin{aligned} P(x_1) &= \int_{-\infty}^{\infty} P(x_1 \mid x_0) P(x_0)\, dx_0 = \alpha \int_{-\infty}^{\infty} e^{-\frac{1}{2}\left(\frac{(x_1-x_0)^2}{\sigma_x^2}\right)} e^{-\frac{1}{2}\left(\frac{(x_0-\mu_0)^2}{\sigma_0^2}\right)} dx_0 \\ &= \alpha \int_{-\infty}^{\infty} e^{-\frac{1}{2}\left(\frac{\sigma_0^2(x_1-x_0)^2+\sigma_x^2(x_0-\mu_0)^2}{\sigma_0^2\sigma_x^2}\right)} dx_0 . \end{aligned}$$

A simple one-dimensional example

This integral looks rather complicated. The key to progress is to notice that the exponent is the sum of two expressions that are *quadratic* in x_0 and hence is itself a quadratic in x_0. A simple trick known as **completing the square** allows the rewriting of any quadratic $ax_0^2 + bx_0 + c$ as the sum of a squared term $a(x_0 - \frac{-b}{2a})^2$ and a residual term $c - \frac{b^2}{4a}$ that is independent of x_0. The residual term can be taken outside the integral, giving us

$$P(x_1) = \alpha\, e^{-\frac{1}{2}\left(c - \frac{b^2}{4a}\right)} \int_{-\infty}^{\infty} e^{-\frac{1}{2}\left(a(x_0 - \frac{-b}{2a})^2\right)}\, dx_0 \,.$$

Now the integral is just the integral of a Gaussian over its full range, which is simply 1. Thus, we are left with only the residual term from the quadratic. Then, we notice that the residual term is a quadratic in x_1; in fact, after simplification, we obtain

$$P(x_1) = \alpha\, e^{-\frac{1}{2}\left(\frac{(x_1 - \mu_0)^2}{\sigma_0^2 + \sigma_x^2}\right)} \,.$$

That is, the one-step predicted distribution is a Gaussian with the same mean μ_0 and a variance equal to the sum of the original variance σ_0^2 and the transition variance σ_x^2.

A simple one-dimensional example

- To complete the update step, we need to condition on the observation at the first time step, namely, z_1. From Equation (15.18), this is given by

$$\begin{aligned} P(x_1 \mid z_1) &= \alpha P(z_1 \mid x_1) P(x_1) \\ &= \alpha e^{-\frac{1}{2}\left(\frac{(z_1 - x_1)^2}{\sigma_z^2}\right)} e^{-\frac{1}{2}\left(\frac{(x_1 - \mu_0)^2}{\sigma_0^2 + \sigma_x^2}\right)} \end{aligned}$$

- Once again, we combine the exponents and complete the square, obtaining

$$P(x_1 \mid z_1) = \alpha e^{-\frac{1}{2}\left(\frac{(x_1 - \frac{(\sigma_0^2 + \sigma_x^2) z_1 + \sigma_z^2 \mu_0}{\sigma_0^2 + \sigma_x^2 + \sigma_z^2})^2}{(\sigma_0^2 + \sigma_x^2)\sigma_z^2 / (\sigma_0^2 + \sigma_x^2 + \sigma_z^2)}\right)} \qquad (15.19)$$

Thus, after one update cycle, we have a new Gaussian distribution for the state variable.

A simple one-dimensional example

- From the Gaussian formula in Equation (15.19), we see that the new mean and standard deviation can be calculated from the old mean and standard deviation as follows:

$$\mu_{t+1} = \frac{(\sigma_t^2 + \sigma_x^2) z_{t+1} + \sigma_z^2 \mu_t}{\sigma_t^2 + \sigma_x^2 + \sigma_z^2} \qquad \text{and} \qquad \sigma_{t+1}^2 = \frac{(\sigma_t^2 + \sigma_x^2)\sigma_z^2}{\sigma_t^2 + \sigma_x^2 + \sigma_z^2} \tag{15.20}$$

- Figure 15.10 shows one update cycle for particular values of the transition and sensor models.

A simple one-dimensional example

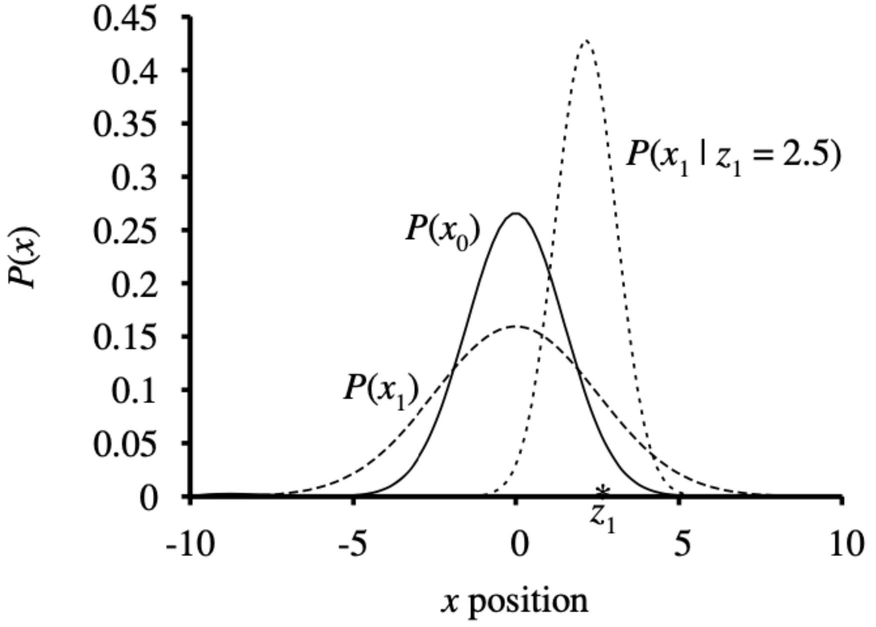


Figure 15.10 Stages in the Kalman filter update cycle for a random walk with a prior given by $\mu_0 = 0.0$ and $\sigma_0 = 1.0$, transition noise given by $\sigma_x = 2.0$, sensor noise given by $\sigma_z = 1.0$, and a first observation $z_1 = 2.5$ (marked on the x-axis). Notice how the prediction $P(x_1)$ is flattened out, relative to $P(x_0)$, by the transition noise. Notice also that the mean of the posterior distribution $P(x_1 \mid z_1)$ is slightly to the left of the observation z_1 because the mean is a weighted average of the prediction and the observation.

A simple one-dimensional example

- Equation (15.20) has some interesting additional properties.
- First, we can interpret the calculation for the new mean μ_{t+1} as simply a weighted mean of the new observation z_{t+1} and the old mean μ_t. If the observation is unreliable, then σ_z^2 is large and we pay more attention to the old mean; if the old mean is unreliable (σ_t^2 is large) or the process is highly unpredictable (σ_x^2 is large), then we pay more attention to the observation.

$$\boxed{\mu_{t+1} = \frac{(\sigma_t^2 + \sigma_x^2) z_{t+1} + \sigma_z^2 \mu_t}{\sigma_t^2 + \sigma_x^2 + \sigma_z^2} \qquad \text{and} \qquad \sigma_{t+1}^2 = \frac{(\sigma_t^2 + \sigma_x^2)\sigma_z^2}{\sigma_t^2 + \sigma_x^2 + \sigma_z^2}}$$

A simple one-dimensional example

- Second, notice that the update for the variance $\sigma_{t+1}{}^2$ is *independent of the observation*. We can therefore compute in advance what the sequence of variance values will be.
- Third, the sequence of variance values converges quickly to a fixed value that depends only on $\sigma_x{}^2$ and $\sigma_z{}^2$, thereby substantially simplifying the subsequent calculations.

$$\boxed{\mu_{t+1} = \frac{(\sigma_t^2 + \sigma_x^2) z_{t+1} + \sigma_z^2 \mu_t}{\sigma_t^2 + \sigma_x^2 + \sigma_z^2} \qquad \text{and} \qquad \sigma_{t+1}^2 = \frac{(\sigma_t^2 + \sigma_x^2)\sigma_z^2}{\sigma_t^2 + \sigma_x^2 + \sigma_z^2}}$$

The general case

- The preceding derivation illustrates the key property of Gaussian distributions that allows Kalman filtering to work: the fact that the exponent is a quadratic form. This is true not just for the univariate case; the full multivariate Gaussian distribution has the form

$$N(\boldsymbol{\mu}, \boldsymbol{\Sigma})(\mathbf{x}) = \alpha\, e^{-\frac{1}{2}\left((\mathbf{x}-\boldsymbol{\mu})^{\top}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)}$$

- Multiplying out the terms in the exponent makes it clear that the exponent is also a quadratic function of the values x_i in $\mathbf{x}$. As in the univariate case, the filtering update preserves the Gaussian nature of the state distribution.

The general case

- Let us first define the general temporal model used with Kalman filtering. Both the transition model and the sensor model allow for a linear transformation with additive Gaussian noise. Thus, we have

$$\begin{aligned} P(\mathbf{x}_{t+1} \mid \mathbf{x}_t) &= N(\mathbf{F}\mathbf{x}_t, \mathbf{\Sigma}_x)(\mathbf{x}_{t+1}) \\ P(\mathbf{z}_t \mid \mathbf{x}_t) &= N(\mathbf{H}\mathbf{x}_t, \mathbf{\Sigma}_z)(\mathbf{z}_t) \ , \end{aligned}$$

- where **F** and $\mathbf{\Sigma}_x$ are matrices describing the linear transition model and transition noise covariance, and **H** and $\mathbf{\Sigma}_z$ are the corresponding matrices for the sensor model.

The general case

- Now the update equations for the mean and covariance are

$$\begin{aligned}\boldsymbol{\mu}_{t+1} &= \mathbf{F}\boldsymbol{\mu}_t + \mathbf{K}_{t+1}(\mathbf{z}_{t+1} - \mathbf{H}\mathbf{F}\boldsymbol{\mu}_t)\\ \boldsymbol{\Sigma}_{t+1} &= (\mathbf{I} - \mathbf{K}_{t+1}\mathbf{H})(\mathbf{F}\boldsymbol{\Sigma}_t\mathbf{F}^\top + \boldsymbol{\Sigma}_x)\end{aligned}$$

  where $\mathbf{K}_{t+1} = (\mathbf{F}\boldsymbol{\Sigma}_t\mathbf{F}^\top + \boldsymbol{\Sigma}_x)\mathbf{H}^\top(\mathbf{H}(\mathbf{F}\boldsymbol{\Sigma}_t\mathbf{F}^\top + \boldsymbol{\Sigma}_x)\mathbf{H}^\top + \boldsymbol{\Sigma}_z)^{-1}$

  is called the **Kalman gain matrix**.

- Consider the update for the mean state estimate $\boldsymbol{\mu}$. The term $\mathbf{F}\boldsymbol{\mu}_t$ is the *predicted* state at t+1, so $\mathbf{H}\mathbf{F}\boldsymbol{\mu}_t$ is the *predicted* observation. Therefore, the term $\mathbf{z}_{t+1} - \mathbf{H}\mathbf{F}\boldsymbol{\mu}_t$ represents the error in the predicted observation. This is multiplied by $\mathbf{K}_{t+1}$ to correct the predicted state; hence, $\mathbf{K}_{t+1}$ is a measure of *how seriously to take the new observation relative to the prediction.*

The general case

- To illustrate these equations at work, we have applied them to the problem of tracking an object moving on the X–Y plane. The state variables are $\mathbf{X} = (X, Y, \underline{X}, \underline{Y})^T$, so $\mathbf{F}$, $\mathbf{\Sigma}_x$, $\mathbf{H}$, and $\mathbf{\Sigma}_z$ are 4×4 matrices.
- Figure 15.11(a) shows the true trajectory, a series of noisy observations, and the trajectory estimated by Kalman filtering, along with the covariances indicated by the one-standard-deviation contours. The filtering process does a good job of tracking the actual motion, and, as expected, the variance quickly reaches a fixed point.

The general case

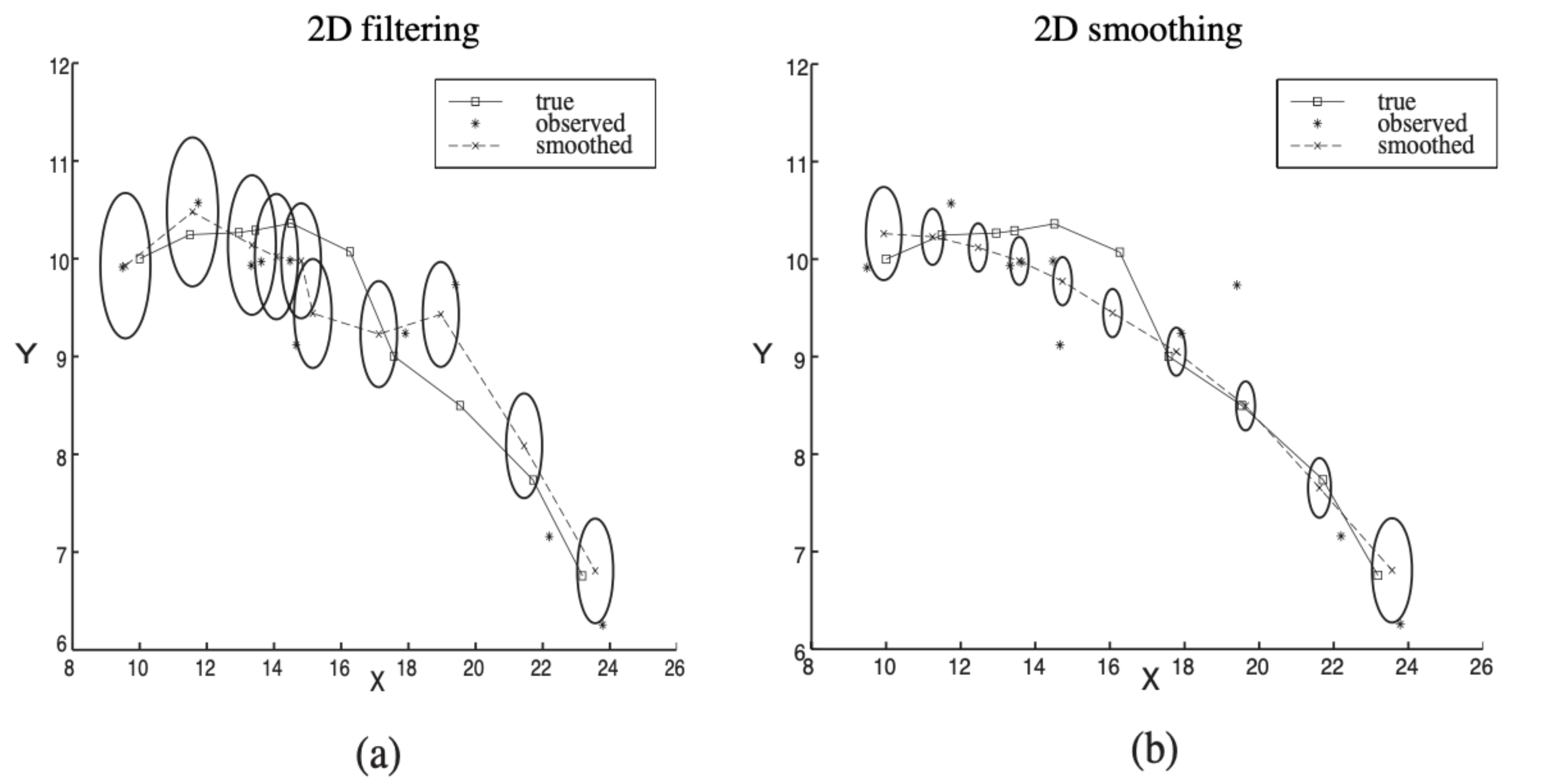


Figure 15.11 (a) Results of Kalman filtering for an object moving on the X–Y plane, showing the true trajectory (left to right), a series of noisy observations, and the trajectory estimated by Kalman filtering. Variance in the position estimate is indicated by the ovals. (b) The results of Kalman smoothing for the same observation sequence.

The general case

- We can also derive equations for smoothing as well as filtering with linear Gaussian models. The smoothing results are shown in Figure 15.11(b). Notice how the variance in the position estimate is sharply reduced, except at the ends of the trajectory, and that the estimated trajectory is much smoother.

Applicability of Kalman filtering

- The "classical" application is in radar tracking of aircraft and missiles. Related applications include acoustic tracking of submarines and ground vehicles and visual tracking of vehicles and people.
- The range of application is much larger than just the tracking of motion: any system characterized by continuous state variables and noisy measurements will do. Such systems include pulp mills, chemical plants, nuclear reactors, plant ecosystems, and national economies.

Keeping Track of Many Objects

- In this section, we see what happens when two or more objects generate the observations. What makes this case different from plain old state estimation is that there is now the possibility of uncertainty about which object generated which observation.
- In the control theory literature, this is the **data association** problem—that is, the problem of associating observation data with the objects that generated them.

Keeping Track of Many Objects

- The data association problem was studied originally in the context of radar tracking, where reflected pulses are detected at fixed time intervals by a rotating radar antenna. At each time step, multiple blips may appear on the screen, but there is no direct observation of which blips at time t belong to which blips at time $t - 1$.
- Figure 15.19(a) shows a simple example with two blips per time step for five steps. Let the two blip locations at time t be e_t^1 and e_t^2. Let us assume, for the time being, that exactly two aircraft, A and B, generated the blips; their true positions are X_t^A and X_t^B. Just to keep things simple, we'll also assume that the each aircraft moves independently according to a known transition model.

Keeping Track of Many Objects

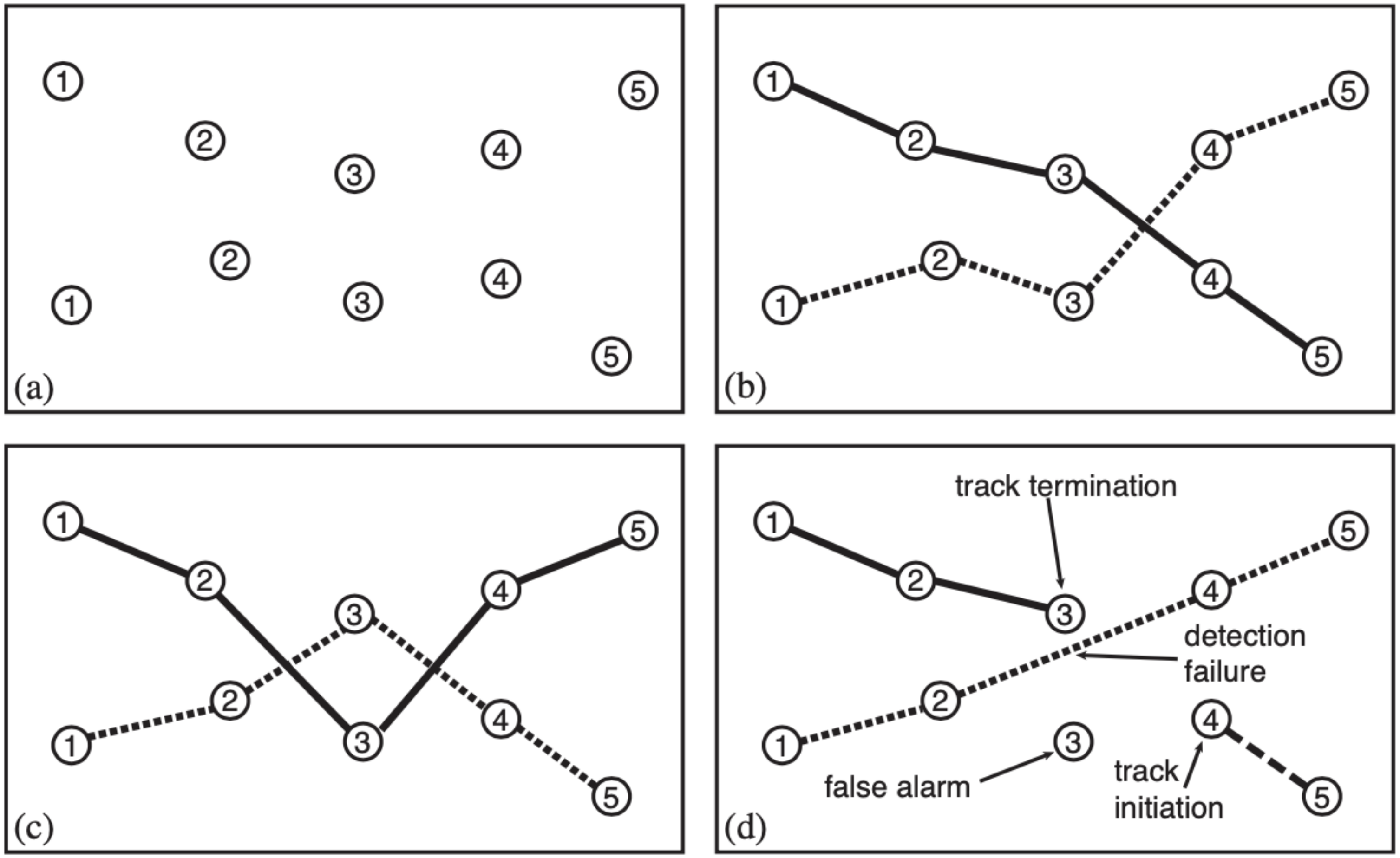


Figure 15.19 (a) Observations made of object locations in 2D space over five time steps. Each observation is labeled with the time step but does not identify the object that produced it. (b–c) Possible hypotheses about the underlying object tracks. (d) A hypothesis for the case in which false alarms, detection failures, and track initiation/termination are possible.

Keeping Track of Many Objects

- Suppose we try to write down the overall probability model for this scenario, just as we did for general temporal processes in Equation (15.3). As usual, the joint distribution factors into contributions for each time step as follows:

$$P(x^A_{0:t}, x^B_{0:t}, e^1_{1:t}, e^2_{1:t}) =$$
$$P(x^A_0)P(x^B_0)\prod_{i=1}^{t} P(x^A_i \mid x^A_{i-1})P(x^B_i \mid x^B_{i-1})\, P(e^1_i, e^2_i \mid x^A_i, x^B_i) \tag{15.24}$$

$$\boxed{\mathbf{P}(\mathbf{X}_{0:t}, \mathbf{E}_{1:t}) = \mathbf{P}(\mathbf{X}_0)\prod_{i=1}^{t} \mathbf{P}(\mathbf{X}_i \mid \mathbf{X}_{i-1})\,\mathbf{P}(\mathbf{E}_i \mid \mathbf{X}_i)}$$

國立成功大學 National Cheng Kung University 1931

Keeping Track of Many Objects

- We would like to factor the observation term $P(e_i^1, e_i^2 \,|\, x_i^A, x_i^B)$ into a product of two terms, one for each object, but this would require knowing which observation was generated by which object. Instead, we have to sum over all possible ways of associating the observations with the objects.
- Some of those ways are shown in Figure 15.19(b–c); in general, for *n* objects and *T* time steps, there are $(n!)^T$ ways of doing it—an awfully large number.

Keeping Track of Many Objects

- We'll write ω_t to denote the one-to-one mapping from objects to observations at time t, with $\omega_t(A)$ and $\omega_t(B)$ denoting the specific observations (1 or 2) that ω_t assigns to A and B. (For n objects, ω_t will have $n!$ possible values; here, $n!=2$.)
- Because the labels "1" and "2" on the observations are assigned arbitrarily, the prior on ω_t is uniform and ω_t is independent of the states of the objects, x_t^A and x_t^B.

Keeping Track of Many Objects

So we can condition the observation term $P(e_i^1, e_i^2 \mid x_i^A, x_i^B)$ on ω_t and then simplify:

$$\begin{aligned} P(e_i^1, e_i^2 \mid x_i^A, x_i^B) &= \sum_{\omega_i} P(e_i^1, e_i^2 \mid x_i^A, x_i^B, \omega_i) P(\omega_i \mid x_i^A, x_i^B) \\ &= \sum_{\omega_i} P(e_i^{\omega_i(A)} \mid x_i^A) P(e_i^{\omega_i(B)} \mid x_i^B) P(\omega_i \mid x_i^A, x_i^B) \\ &= \frac{1}{2} \sum_{\omega_i} P(e_i^{\omega_i(A)} \mid x_i^A) P(e_i^{\omega_i(B)} \mid x_i^B) \,. \end{aligned}$$

Plugging this into Equation (15.24), we get an expression that is only in terms of transition and sensor models for individual objects and observations.

國立成功大學 National Cheng Kung University 1931

Keeping Track of Many Objects

- As for all probability models, inference means summing out the variables other than the query and the evidence. For filtering in HMMs and DBNs, we were able to sum out the state variables from 1 to $t-1$ by a simple dynamic programming trick; for Kalman filters, we took advantage of special properties of Gaussians.
- For data association, we are less fortunate. There is no (known) efficient exact algorithm, for the same reason that there is none for the switching Kalman filter.

Keeping Track of Many Objects

- As a result of the complexity of exact inference, many different approximate methods have been used. The simplest approach is to choose a single "best" assignment at each time step, given the predicted positions of the objects at the current time step. This assignment associates observations with objects and enables the track of each object to be updated and a prediction made for the next time step.
- For choosing the "best" assignment, it is common to use the so-called **nearest-neighbor filter**, which repeatedly chooses the closest pairing of predicted position and observation and adds that pairing to the assignment.

Keeping Track of Many Objects

- The nearest-neighbor filter works well when the objects are well separated in state space and the prediction uncertainty and observation error are small.
- When there is more uncertainty as to the correct assignment, a better approach is to choose the assignment that maximizes the joint probability of the current observations given the predicted positions. This can be done very efficiently using the **Hungarian algorithm**, even though there are *n*! assignments to choose from.

Keeping Track of Many Objects

- Any method that commits to a single best assignment at each time step fails miserably under more difficult conditions. In particular, if the algorithm commits to an incorrect assignment, the prediction at the next time step may be significantly wrong, leading to more incorrect assignments, and so on.
- Two modern approaches turn out to be much more effective. A **particle filtering** algorithm for data association works by maintaining a large collection of possible current assignments.

Keeping Track of Many Objects

- An **MCMC** algorithm explores the space of assignment histories—for example, Figure 15.19(b–c) might be states in the MCMC state space—and can change its mind about previous assignment decisions. Current MCMC data association methods can handle many hundreds of objects in real time while giving a good approximation to the true posterior distributions.

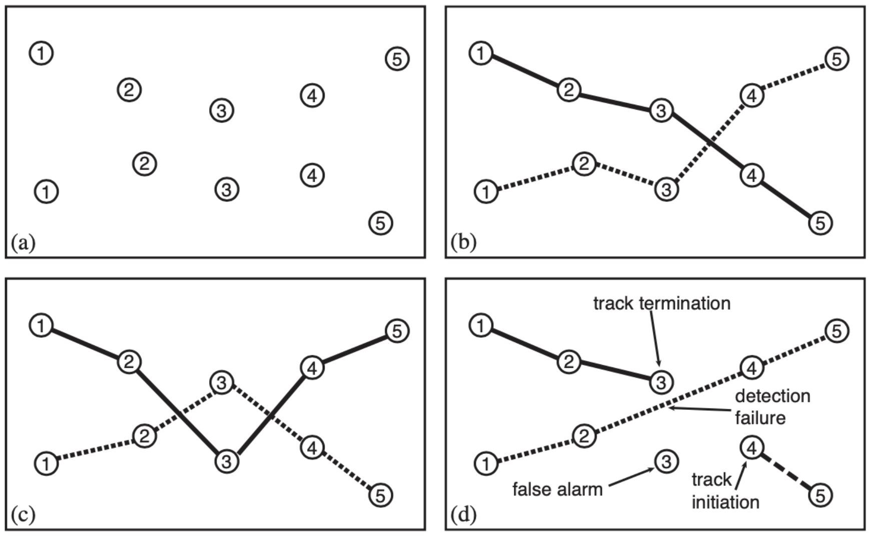


Figure 15.19 (a) Observations made of object locations in 2D space over five time steps. Each observation is labeled with the time step but does not identify the object that produced it. (b–c) Possible hypotheses about the underlying object tracks. (d) A hypothesis for the case in which false alarms, detection failures, and track initiation/termination are possible.