Unit IV – PROBABILISTIC REASONING: Representing Knowledge in an uncertain domain, Semantics of Bayesian networks, Probabilistic reasoning over time, Time and uncertainty, Inference in temporal models, Hidden Markov models, Kalman Filter.

"To build efficient network models to reason under uncertainty according to the laws of probability theory

Extends the basic ideas of Bayesian networks to more expressive formal languages for defining probability models.

4.1 Representing Knowledge in uncertain Domain:

- Specifying probabilities for possible worlds one by one is unnatural and
- Tedious independence and conditional independence relationships among
- variables can greatly reduce the number of probabilities that need to be specified in order to define the full joint distribution
- A data structure called a Bayesian network can be used to represent the dependencies among variables. Bayesian networks can represent essentially any full joint probability distribution and, in many cases, can do so very concisely

A Bayesian network is a directed graph in which each node is annotated with quantitative probability information. Full specification is:

1. Each node corresponds to a random variable, which may be discrete or continuous.
2. Directed links or arrows connect pairs of nodes. If there is an arrow from node X to node Y, X is said to be a *parent* of Y. The graph has no directed cycles and hence is a directed acyclic graph, or DAG.
3. Each node X_i has associated probability information $\theta(X_i|Parents(X_i))$ that quantifies the effect of the parents on the node using a finite number of **parameters**.

- The topology of the network—the set of nodes and links—specifies the conditional independence relationships that hold in the domain, in a way that will be made precise shortly
- The intuitive meaning of an arrow is typically that X has a direct influence on ,which suggests that causes should be parents of effects
- Once the topology of the Bayes net is laid out, we need only specify the local probability information for each variable, in the form of a conditional distribution given its parents.
- The full joint distribution for all the variables is defined by the topology and the local probability information.

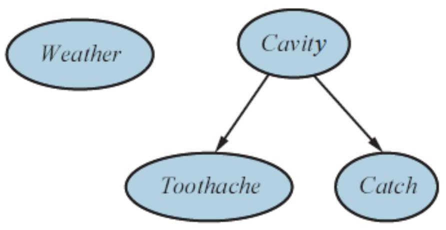
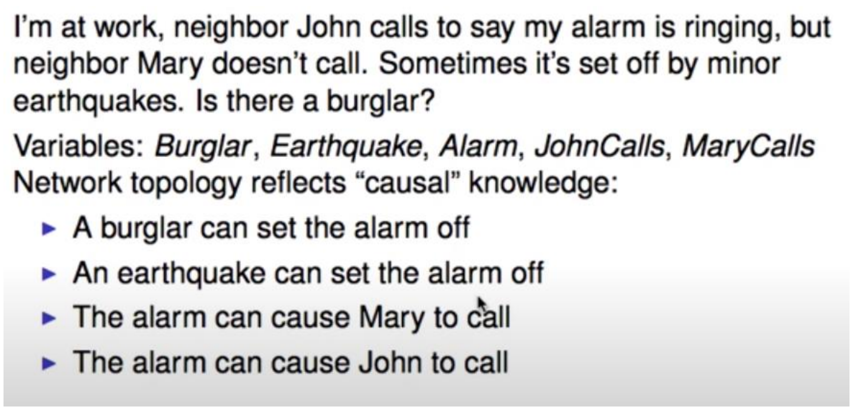


Figure 13.1 A simple Bayesian network in which *Weather* is independent of the other three variables and *Toothache* and *Catch* are conditionally independent, given *Cavity*.

Recall the simple world described in Chapter 12, consisting of the variables *Toothache*, *Cavity*, *Catch*, and *Weather*. We argued that *Weather* is independent of the other variables; furthermore, we argued that *Toothache* and *Catch* are conditionally independent, given *Cavity*. These relationships are represented by the Bayes net structure shown in Figure 13.1. Formally, the conditional independence of *Toothache* and *Catch*, given *Cavity*, is indicated by the *absence* of a link between *Toothache* and *Catch*. Intuitively, the network represents the fact that *Cavity* is a direct cause of *Toothache* and *Catch*, whereas no direct causal relationship exists between *Toothache* and *Catch*.

Bayesian Network Example



I'm at work, neighbor John calls to say my alarm is ringing, but neighbor Mary doesn't call. Sometimes it's set off by minor earthquakes. Is there a burglar?

Variables: *Burglar*, *Earthquake*, *Alarm*, *JohnCalls*, *MaryCalls*
Network topology reflects "causal" knowledge:

- A burglar can set the alarm off
- An earthquake can set the alarm off
- The alarm can cause Mary to call
- The alarm can cause John to call

Bayesian Network Example

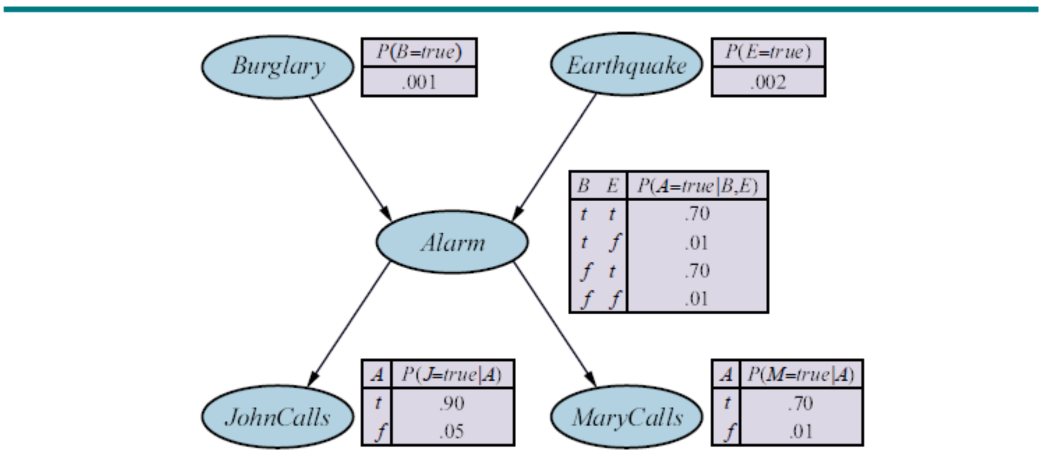
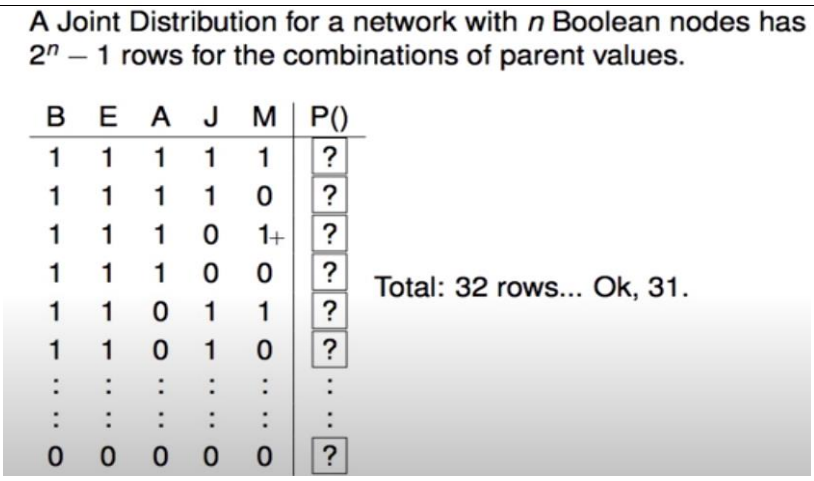


Figure 13.2 A typical Bayesian network, showing both the topology and the conditional probability tables (CPTs). In the CPTs, the letters B, E, A, J, and M stand for *Burglary*, *Earthquake*, *Alarm*, *JohnCalls*, and *MaryCalls*, respectively.


A Joint Distribution for a network with n Boolean nodes has $2^n - 1$ rows for the combinations of parent values.

B	E	A	J	M	P()
1	1	1	1	1	?
1	1	1	1	0	?
1	1	1	0	1	?
1	1	1	0	0	?
1	1	0	1	1	?
1	1	0	1	0	?
:	:	:	:	:	:
:	:	:	:	:	:
0	0	0	0	0	?

Total: 32 rows... Ok, 31.

4.2 Semantics of Bayesian Networks:

- Conditional independence relations in Bayesian networks
- Efficient Representation of Conditional Distributions
- Bayesian nets with continuous variables
- Case study: Car insurance
- The syntax of a Bayes net consists of a directed acyclic graph with some local probability information attached to each node.
- The semantics defines how the syntax corresponds to a joint distribution over the variables of the network.

Assume that the Bayes net contains n variables, $X_1, \ldots, X_n$. A generic entry in the joint distribution is then $P(X_1 = x_1 \wedge \ldots \wedge X_n = x_n)$, or $P(x_1, \ldots, x_n)$ for short. The semantics of Bayes nets defines each entry in the joint distribution as follows:

(13.1)

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} \theta(x_i | parents(X_i)),$$

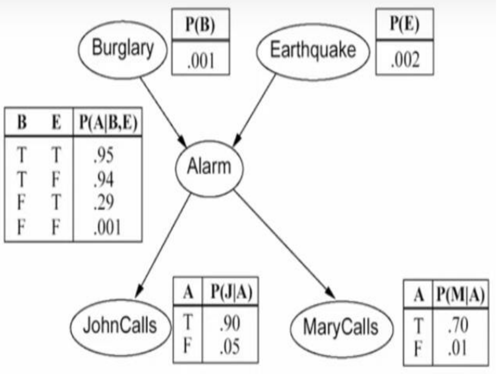
where $parents(X_i)$ denotes the values of $Parents(X_i)$ that appear in $x_1, \ldots, x_n$. Thus, each entry in the joint distribution is represented by the product of the appropriate elements of the local conditional distributions in the Bayes net.

To illustrate this, we can calculate the probability that the alarm has sounded, but neither a burglary nor an earthquake has occurred, and both John and Mary call. We simply multiply the relevant entries from the local conditional distributions (abbreviating the variable names):

$$\begin{aligned} P(j, m, a, \neg b, \neg e) &= P(j|a)P(m|a)P(a|\neg b \wedge \neg e)P(\neg b)P(\neg e) \\ &= 0.90 \times 0.70 \times 0.01 \times 0.999 \times 0.998 = 0.00628. \end{aligned}$$

Section 12.3 explained that the full joint distribution can be used to answer any query about the domain. If a Bayes net is a representation of the joint distribution, then it too can be used to answer any query, by summing all the relevant joint probability values, each calculated by multiplying probabilities from the local conditional distributions.

1. What is the probability that the alarm has sounded but neither a burglary nor an earthquake has occurred, and both John and Merry call?



P(B)
.001

P(E)
.002

B	E	P(A|B,E)
T	T	.95
T	F	.94
F	T	.29
F	F	.001

A	P(J|A)
T	.90
F	.05

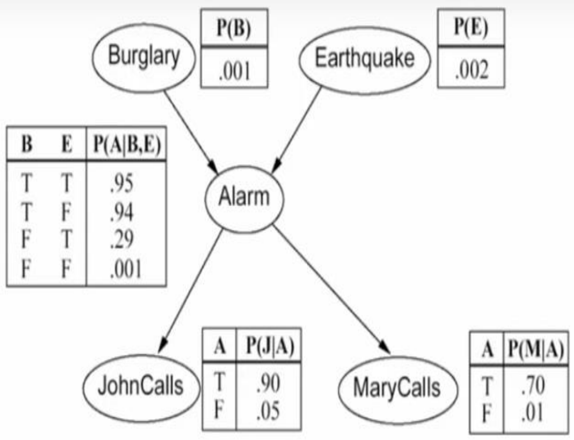
A	P(M|A)
T	.70
F	.01

Solution:

$$P(j \wedge m \wedge a \wedge \neg b \wedge \neg e) = P(j \mid a)\, P(m \mid a)\, P(a \mid \neg b, \neg e)\, P(\neg b)\, P(\neg e)$$

$$= 0.90 \times 0.70 \times 0.001 \times 0.999 \times 0.998$$

2. What is the probability that John call?



P(B)
.001

P(E)
.002

B	E	P(A|B,E)
T	T	.95
T	F	.94
F	T	.29
F	F	.001

A	P(J|A)
T	.90
F	.05

A	P(M|A)
T	.70
F	.01

Solution:

$$P(j) = P(j \mid a)\, P(a) + P(j \mid \neg a)\, P(\neg a)$$

$$= P(j|a)\{P(a|b,e)*P(b,e)+P(a|\neg b,e)*P(\neg b,e)+P(a|b,\neg e)*P(b,\neg e)+P(a|\neg b,\neg e)*P(\neg b,\neg e)\}$$

$$+ P(j|\neg a)\{P(\neg a|b,e)*P(b,e)+P(\neg a|\neg b,e)* P(\neg b,e)+P(\neg a|b,\neg e)* P(b,\neg e)+P(\neg a|\neg b, \neg e)* P(\neg b, \neg e)\}$$

conditional probabilities can be computed from the joint distribution as follows:

$$\begin{aligned} P(x_i|parents(X_i)) &\equiv \frac{P(x_i, parents(X_i))}{P(parents(X_i))} \\ &= \frac{\sum_{\mathbf{y}} P(x_i, parents(X_i), \mathbf{y})}{\sum x'_{i}, \mathbf{y} P(x'_i, parents(X_i), \mathbf{y})} \end{aligned}$$

where $\mathbf{y}$ represents the values of all variables other than X_i and its parents. From this last line one can prove that $P(x_i|parents(X_i)) = \theta(x_i|parents(X_i))$ (Exercise 13.CPTE). Hence, we can rewrite Equation (13.1) as

(13.2)

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i|parents(X_i)).$$

- **A Method for Constructing Bayesian Networks**

$$P(x_1,\ldots,x_n) = \prod_{i=1}^{n} P(x_i|parents(X_i)).$$

- ❖ The above equation implies certain conditional independence relationships that can be used to guide the knowledge engineer in constructing the topology of the network.
- ❖ First, we rewrite the entries in the joint distribution in terms of conditional probability, using the product rule

$$P(x_1,\ldots,x_n) = P(x_n|x_{n-1},\ldots,x_1)P(x_{n-1},\ldots,x_1).$$

Then we repeat the process, reducing each joint probability to a conditional probability and a joint probability on a smaller set of variables. We end up with one big product:

$$\begin{aligned} P(x_1,\ldots,x_n) &= P(x_n|x_{n-1},\ldots,x_1)P(x_{n-1}|x_{n-2},\ldots,x_1)\cdots P(x_2|x_1)P(x_1) \\ &= \prod_{i=1}^{n} P(x_i|x_{i-1},\ldots,x_1). \end{aligned}$$

- ❖ This identity is called the chain rule.
- ❖ It holds for any set of random variables. Comparing it with previous we see that the specification of the joint distribution is equivalent to the general assertion that, for every variable Xi in the network
- (equ.13.3)--------- >

$$\mathbf{P}(X_i|X_{i-1},\ldots,X_1) = \mathbf{P}(X_i|Parents(X_i)),$$

-

provided that $Parents(X_i) \subseteq \{X_{i-1},\ldots,X_1\}$. This last condition is satisfied by numbering the nodes in **topological order**—that is, in any order consistent with the directed graph structure. For example, the nodes in Figure 13.2 could be ordered B,E,A,J,M; E,B,A,M,J; and so on.

- ❖ The Bayesian network is a correct representation of the domain only if each node is conditionally independent of its other predecessors in the node ordering, given its parents. We can satisfy this condition with this methodology:
- ❖ NODES: First determine the set of variables that are required to model the domain. Now order them, {X1,…,Xn} . Any order will work, but the resulting network will be more compact if the variables are ordered such that causes precede effects.
- **LINKS: For to do:** Choose a minimal set of parents for from , such that Equation (13.3) is satisfied.
  - For each parent insert a link from the parent to Xi .
  - CPTs: Write down the conditional probability table P(Xi|Parents(Xi)

The following conditional independence statement holds:

$$\mathbf{P}(MaryCalls|JohnCalls,Alarm,Earthquake,Burglary) = \mathbf{P}(MaryCalls|Alarm).$$

- Thus, Alarm will be the only parent node for Mary Calls
- Because each node is connected only to earlier nodes, this construction method guarantees that the network is acyclic.
- Another important property of Bayes nets is that they contain no redundant probability values. If there is no redundancy, then there is no chance for inconsistency: it is impossible for the knowledge engineer or domain expert to create a Bayesian network that violates the axioms of probability.

❖ Compactness and node ordering

- The compactness of Bayesian networks is an example of a general property of locally structured (also called sparse) systems.
- In a locally structured system, each subcomponent interacts directly with only a bounded number of other components, regardless of the total number of components
- In the case of Bayes nets, it is reasonable to suppose that in most domains each random variable is directly influenced by at most others, for some constant.
- If we assume Boolean variables for simplicity, then the amount of information needed to specify each conditional probability table will be at most 2k numbers, and the complete network can be specified by 2k..n numbers.
- In contrast, the joint distribution contains 2n numbers.
- To make this concrete, suppose we have nodes, each with five parents (k=5). Then the Bayesian network requires 960 numbers, but the full joint distribution requires over a billion.

Suppose we decide to add the nodes in the order Mary Calls John Calls , Alarm, Burglar, Earthquake. We then get the somewhat more complicated network shown in Figure 13.3(a) . The process goes as follows:

- Adding $MaryCalls$: No parents.
- Adding $JohnCalls$: If Mary calls, that probably means the alarm has gone off, which makes it more likely that John calls. Therefore, $JohnCalls$ needs $MaryCalls$ as a parent.
- Adding *Alarm*: Clearly, if both call, it is more likely that the alarm has gone off than if just one or neither calls, so we need both $MaryCalls$ and $JohnCalls$ as parents.

- Adding *Burglar*: If we know the alarm state, then the call from John or Mary might give us information about our phone ringing or Mary's music, but not about burglary:

$$\mathbf{P}(Burglary|Alarm, JohnCalls, MaryCalls) = \mathbf{P}(Burglary|Alarm).$$

  Hence we need just *Alarm* as parent.
- Adding *Earthquake*: If the alarm is on, it is more likely that there has been an earthquake. (The alarm is an earthquake detector of sorts.) But if we know that there has been a burglary, then that explains the alarm, and the probability of an earthquake would be only slightly above normal. Hence, we need both *Alarm* and *Burglar* as parents.

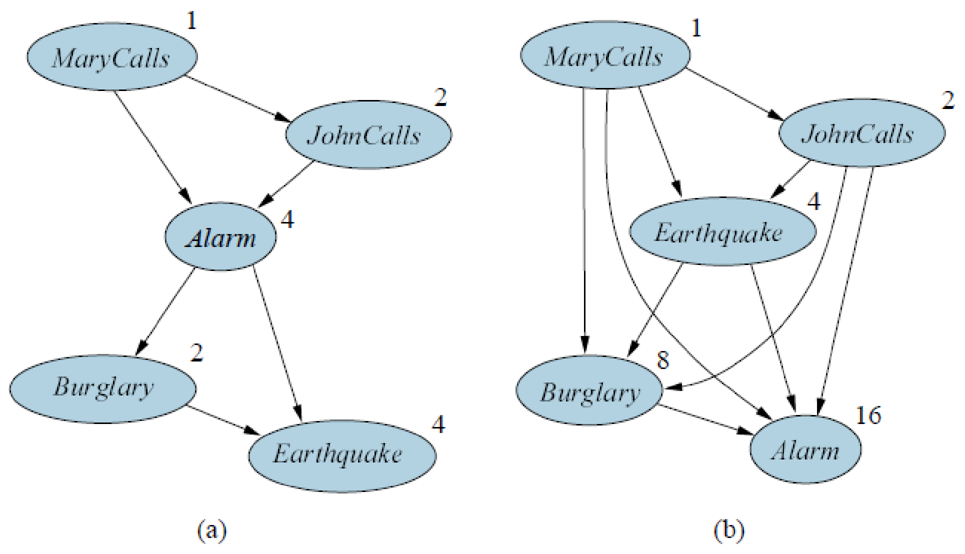


Figure 13.3 Network structure and number of parameters depends on order of introduction. (a) The structure obtained with ordering M, J, A, B, E. (b) The structure obtained with M, J, E, B, A. Each node is annotated with the number of parameters required; 13 in all for (a) and 31 for (b). In Figure ??, only 10 parameters were required.

❖ Conditional independence relations in Bayesian networks:

From the semantics of Bayes nets we can derive a number of conditional independence properties.

- It is also possible to prove the more general "non-descendants" property that:
- Each variable is conditionally independent of its non-descendants, given its parents.
- The variable JohnCalls is independent of Burglar, Earthquake,and MaryCalls given the value of Alarm. The definition is illustrated in figure 13.4(a):

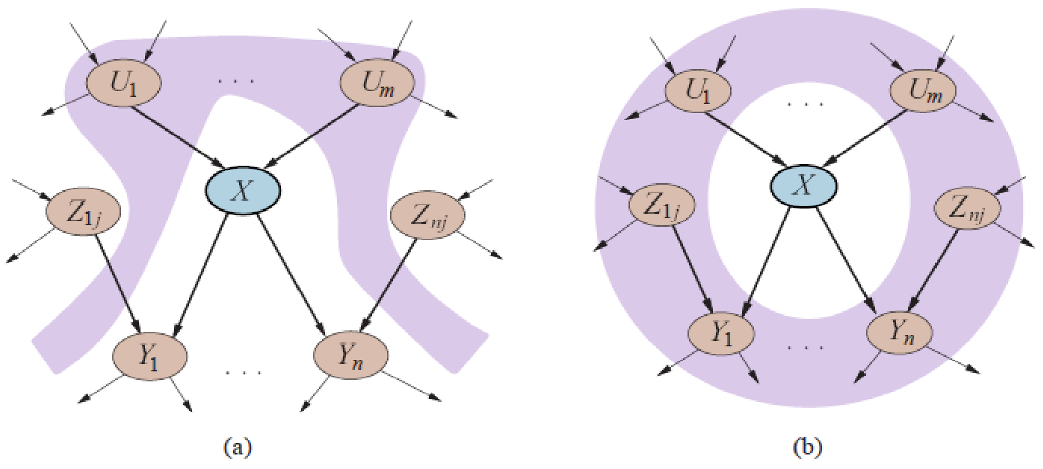


Figure 13.4 (a) A node X is conditionally independent of its non-descendants (e.g., the Z_{ij}s) given its parents (the U_is shown in the gray area). (b) A node X is conditionally independent of all other nodes in the network given its Markov blanket (the gray area).

- One can view the semantics of Bayes nets in a different way: instead of defining the full joint distribution as the product of conditional distributions, the network defines a set of conditional independence properties.
- The full joint distribution can be derived from those properties.
- Another important independence property is implied by the non-descendants property:

- A variable is conditionally independent of all other nodes in the network, given its parents, children, and children's parents—that is, given its Markov blanket.
- For example, the variable Burglary is independent of JohnCalls and MaryCalls, given Alarm and Earthquake
- The most general conditional independence question one might ask in a Bayes net is whether a set of nodes X is conditionally independent of another set Y, given a third set Z .
- This can be determined efficiently by examining the Bayes net to see whether Z d-separates X and Y . (D-Separation)The process works as follows:
- 1.Consider just the ancestral subgraph consisting of X, Y , Z , and their ancestors.
- 2.Add links between any unlinked pair of nodes that share a common child; now we
- have the so-called moral graph.
- 3.Replace all directed links by undirected links.
- 4. If Z blocks all paths between X and Y in the resulting graph, then Z d-separates X and Y . In that case X, is conditionally independent of Y , given Z . Otherwise, the original Bayes net does not require conditional independence.
- In brief, then, d-separation means separation in the undirected, moralized, ancestral subgraph.
- Applying the definition to the burglary network we can deduce that Burglar and Earthquake are independent given the empty set (i.e., they are absolutely
- independent); that they are not necessarily conditionally independent given Alarm; and that and are conditionally independent given Alarm.
- Notice also that the Markov blanket property follows directly from the d-separation property, since a variable's Markov blanket d-separates it from all other variables.
- Efficient Representation of Conditional Distributions
- A worst-case scenario in which the relationship between the parents and the child is completely arbitrary CPT for a node requires up to O(2k) numbers
- Usually, such relationships are describable by a canonical distribution that fits some standard pattern
- The simplest example is provided by deterministic nodes. A deterministic node has its value specified exactly by the values of its parents, with no uncertainty.
- Many Bayes net systems allow the user to specify deterministic functions using a general-purpose programming language; this makes it possible to include complex elements such as global climate models or power-grid simulators within a probabilistic model.
- Another important pattern that occurs often in practice is context-specific independence or CSI.
- A conditional distribution exhibits CSI if a variable is conditionally independent of some of its parents given certain values of others

- A worst-case scenario in which the relationship between the parents and the child is completely arbitrary CPT for a node requires up to O(2k) numbers
- Usually, such relationships are describable by a canonical distribution that fits some standard pattern
- The simplest example is provided by deterministic nodes. A deterministic node has its value specified exactly by the values of its parents, with no uncertainty.
- Many Bayes net systems allow the user to specify deterministic functions using a general-purpose programming language; this makes it possible to include complex elements such as global climate models or power-grid simulators within a probabilistic model.

- Another important pattern that occurs often in practice is context-specific independence or CSI.
- A conditional distribution exhibits CSI if a variable is conditionally independent of some of its parents given certain values of others.

For example, let's suppose that the *Damage* to your car occurring during a given period of time depends on the *Ruggedness* of your car and whether or not an *Accident* occurred in that period. Clearly, if *Accident* is false, then the *Damage*, if any, doesn't depend on the *Ruggedness* of your car. (There might be vandalism damage to the car's paintwork or windows, but we'll assume all cars are equally subject to such damage.) We say that *Damage* is context-specifically independent of *Ruggedness* given $Accident = false$. Bayes net systems often implement CSI using an if-then-else syntax for specifying conditional distributions; for example, one might write

$$\mathbf{P}(Damage|Ruggedness, Accident) = \\ \mathbf{if}(Accident = false)\ \mathbf{then}\ d_1\ \mathbf{else}\ d_2(Ruggedness)$$

where d_1 and d_2 represent arbitrary distributions. As with determinism, the presence of CSI in a network may facilitate efficient inference.

- Uncertain relationships can often be characterized by so-called noisy logical relationships.
- The standard example is the noisy-OR relation, which is a generalization of the logical OR.
- In propositional logic, we might say that Fever is true if and only if Cold, Flu or Malaria are true.
- The noisy-OR model allows for uncertainty about the ability of each parent to cause the child to be true—the causal relationship between parent and child may be inhibited, and so a patient could have a cold, but not exhibit a fever.
- The model makes two assumptions. First, it assumes that all the possible causes are listed. (If some are missing, we can always add a so-called leak node that covers "miscellaneous causes.")
- Second, it assumes that inhibition of each parent is independent of inhibition of
- any other parents: for example, whatever inhibits Malaria from causing a fever is independent of whatever inhibits Flu from causing a fever
- Let us suppose these individual inhibition probabilities are as follows:

$$q_{\text{cold}} = P(\neg fever|cold, \neg flu, \neg malaria) = 0.6,$$
$$q_{\text{flu}} = P(\neg fever|\neg cold, flu, \neg malaria) = 0.2,$$
$$q_{\text{malaria}} = P(\neg fever|\neg cold, \neg flu, malaria) = 0.1.$$

Cold	*Flu*	*Malaria*	$P(fever \mid \cdot)$	$P(\neg fever \mid \cdot)$
f	*f*	*f*	0.0	1.0
f	*f*	*t*	0.9	**0.1**
f	*t*	*f*	0.8	**0.2**
f	*t*	*t*	0.98	$0.02 = 0.2 \times 0.1$
t	*f*	*f*	0.4	**0.6**
t	*f*	*t*	0.94	$0.06 = 0.6 \times 0.1$
t	*t*	*f*	0.88	$0.12 = 0.6 \times 0.2$
t	*t*	*t*	0.988	$0.012 = 0.6 \times 0.2 \times 0.1$

Figure 13.5 A complete conditional probability table for ¶$(Fever \mid Cold, Flu, Malaria)$, assuming a noisy-OR model with the the three q-values shown in bold.

❖ Bayesian nets with continuous variables

- Many real-world problems involve continuous quantities, such as height, mass, temperature, and money
- One way to handle continuous variables is with discretization—that is, dividing up the possible values into a fixed set of intervals.
- For example, temperatures could be divided into three categories:

$(< 0^\circ\mathrm{C})$, $(0^\circ\mathrm{C} - 100^\circ\mathrm{C})$, and $(> 100^\circ\mathrm{C})$.

Another approach is to define a continuous variable using one of the standard families of probability density functions

For example, a Gaussian (or normal) distribution $N(x; \mu, \sigma^2)$ is specified by just two parameters, the mean μ and the variance σ^2. Yet another solution—sometimes called a **nonparametric** representation—is to define the conditional distribution implicitly with a collection of instances, each containing specific values of the parent and child variables.

- ➢ A network with both discrete and continuous variables is called a hybrid Bayesian network.
- ➢ To specify a hybrid network, we have to specify two new kinds of distributions: the
- ➢ conditional distribution for a continuous variable given discrete or continuous parents; and the conditional distribution for a discrete variable given continuous parents.
- ➢ Consider the simple example in Figure 13.6 , in which a customer buys some fruit depending on its cost, which depends in turn on the size of the harvest and whether the government’s subsidy scheme is operating.
- ➢ The variable Cost is continuous and has continuous and discrete parents; the variable Buys is discrete and has a continuous parent.

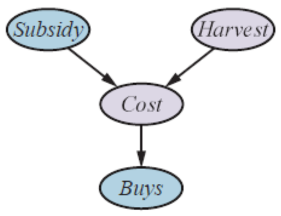


Figure 13.6 A simple network with discrete variables (*Subsidy* and *Buys*) and continuous variables (*Harvest* and *Cost*).

- ➢ Discrete variables are countable in a finite amount of time.
- ➢ Continuous Variables would (literally) take forever to count.
- ➢ In fact, you would get to “forever” and never finish counting them.

For the *Cost* variable, we need to specify $\mathbf{P}(Cost|Harvest,Subsidy)$. The discrete parent is handled by enumeration—that is, by specifying both $\mathbf{P}(Cost|Harvest,subsidy)$ and $\mathbf{P}(Cost|Harvest,\neg subsidy)$. To handle *Harvest*, we specify how the distribution over the cost c depends on the continuous value h of *Harvest*. In other words, we specify the *parameters* of the cost distribution as a function of h. The most common choice is the **linear–Gaussian** conditional distribution, in which the child has a Gaussian distribution whose mean μ varies linearly with the value of the parent and whose standard deviation σ is fixed. We need two distributions, one for *subsidy* and one for $\neg$*subsidy*, with different parameters:

$$P(c|h,subsidy) = N(c;a_t h + b_t,\sigma_t^2) = \frac{1}{\sigma_t\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{c-(a_t h+b_t)}{\sigma_t}\right)^2}$$

$$P(c|h,\neg subsidy) = N(c;a_f h + b_f,\sigma_f^2) = \frac{1}{\sigma_f\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{c-(a_f h+b_f)}{\sigma_f}\right)^2}.$$

Linear–Gaussian

For this example, then, the conditional distribution for *Cost* is specified by naming the linear–Gaussian distribution and providing the parameters a_t, b_t, σ_t, a_f, b_f, and σ_f. Figures 13.7(a) and (b) show these two relationships. Notice that in each case the slope of c versus h is negative, because cost decreases as the harvest size increases. (Of course, the assumption of linearity implies that the cost becomes negative at some point; the linear model is reasonable only if the harvest size is limited to a narrow range.) Figure 13.7(c) shows the distribution $P(c|h)$, averaging over the two possible values of *Subsidy* and assuming that each has prior probability 0.5. This shows that even with very simple models, quite interesting distributions can be represented.

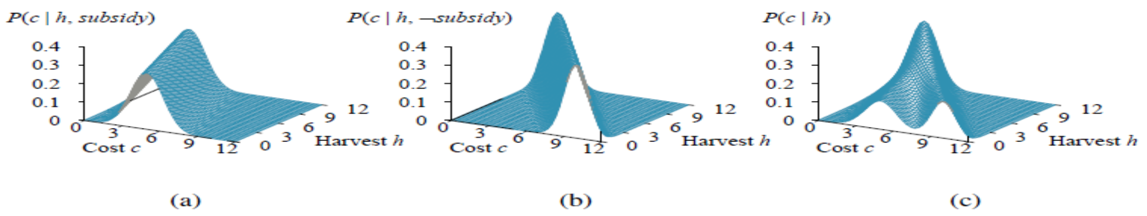
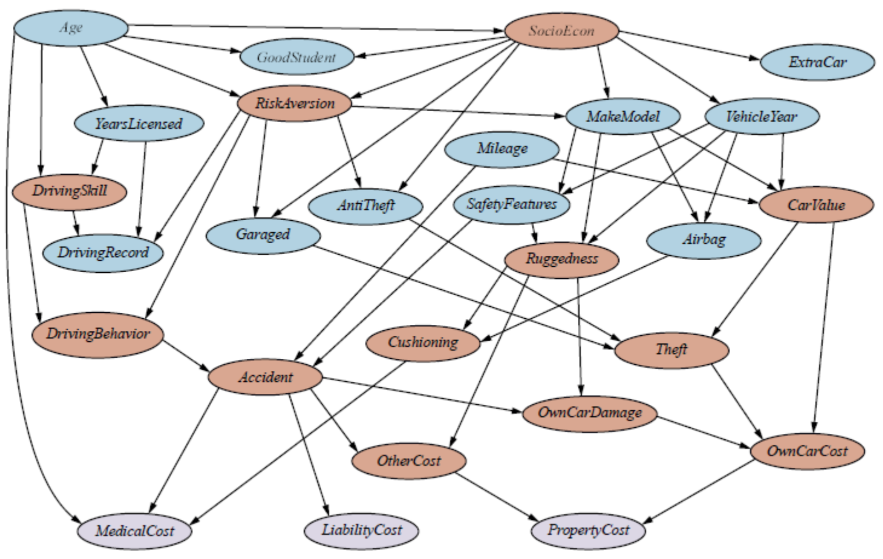


Figure 13.7 The graphs in (a) and (b) show the probability distribution over *Cost* as a function of *Harvest* size, with *Subsidy* true and false, respectively. Graph (c) shows the distribution $P(Cost \mid Harvest)$, obtained by summing over the two subsidy cases.

❖ Case Study – Car Insurance

- A car insurance company receives an application from an individual to insure a specific vehicle and must decide on the to appropriate annual premium to charge, based on the anticipated claims it will pay out for this applicant
- The task is to build a Bayes net that captures the causal structure of the domain and gives an accurate, well-calibrated distribution over the output variables given the evidence available from the application form.
- The Bayes net will include hidden variables that are neither input nor output variables, but are essential for structuring the network so that it is reasonably sparse with a manageable number of parameters



Variables shaded brown are hidden variables

Claims to be paid out shaded lavender

-MedicalCost

Liability Cost

PropertyCost

Bayesian network for evaluating car insurance applications

❖ **Input Information:**

- About the applicant:Age; YearsLicensed, DrivingRecord, GoodStudent
- About the vehicle:MakeModel ,VehicleYear, Airbag, SafetyFeatures(anti-lock braking and collision warning)
- About the driving situation:Mileage(Annual), Garaged
- The key hidden variables are whether or not a Theft or Accident will occur in the next time period. Cannot ask customer,need to be predicted from insurers previous experience(information)

❖ **Casual factors leading to Theft**

- MakeModel (VehicleYear,Mileage),CarValue,AntiTheft

❖ **Another hidden variableSocioEcon**

❖ **Influences**

❖ **MakeModel, VehicleYear, ExtraCar and GoodStudent**

- For insurance company important hidden variable is RiskAversion
  "symptoms" include the applicant's choice of whether the vehicle is Garaged

  and has AntiTheft devices and SafetyFeatures.

- **DrivingBehavior** key to predict accidents
- **Influenced by:**RiskAversion, DrivingSkill,Age,YearLicensed,DrivingRecord
- **If Accident occurs:cost involved**
  - MedicalCost (Age,Cushioning,Ruggedness,Airbag
  - LiabilityCost (Ruggedness and CarValue)

    - PropertyCost(Ruggedness and CarValue)

- **Ranges and Conditional Distribution:**
    - Predict Continuous and Discrete variable
    - Continuous variable eg:Ruggedness (can hold values ranging from 0 to 1)
    - Continuous variables provide more precision, but they make exact inference impossible except in a few special cases
    - A discrete variable with many possible values can make it tedious to fill in the correspondingly large conditional probability tables and makes exact inference more expensive unless the variable's value is always observed
    - For example, MakeModel in a real system would have thousands of possible values, and this causes its child CarValue to have an enormous CPT
- **How to do inference in the network to make predictions?**

- Inference methods in Bayesian Network:
    - Inference by Enumeration
    - Variable Elimination Algorithm
- Each inference method used will be evaluated on the insurance net to measure the time and space requirements of the method.

4.3 Probabilistic Reasoning Over Time

4.3.1 Time and Uncertainty

4.3.2 Inference in Temporal Models

4.3.3 Hidden Markov Model

4.3.4 Kalman Filter

To interpret the present, understand the past, and perhaps predict the future, even when very little is crystal clear.
Agents in partially observable environments must be able to keep track of the current state, agent maintains a

- Belief state that represents which states of the world are currently possible.
- Transition model, the agent can predict how the world might evolve in the next time step.
- Sensor model, the agent can update the belief state.
- Probability theory can be used to quantify the degree of belief in elements of the belief state.
- Changing world is modeled using a variable for each aspect of the world state at each point in time.
- The transition and sensor models may be uncertain:
    - transition model describes the probability distribution of the variables at time given the state of the world at past times.
    - sensor model describes the probability of each percept at time , given the current state of the world.
- Three specific kinds of models: (Temporal)
    - Hidden Markov Model
    - Kalman Filters
    - ynamic Bayesian Networks
    -

4.3.1Time and Uncertainty

- ❖ **States and observations**
- ❖ **Transition and sensor models**

- The techniques discussed for probabilistic reasoning in the context of static worlds, in which each random variable has a single fixed value. Example- repairing a car – Broken remains broken,infer state of car from evidence which is fixed
- Example -treating a diabetic patient evidence such as recent insulin doses, food intake, blood sugar measurements, and other physical signs. (dynamic)
- Task is to assess the current state of the patient, including the actual blood sugar level and insulin level.
- Given this information, decision can be made about the patient's food intake and insulin dose.
- Dynamic aspects of the problem are essential.
- Blood sugar levels and measurements thereof can change rapidly over time, depending on recent food intake and insulin doses, metabolic activity, the time of day, and so on
- To assess the current state from the history of evidence and to predict the outcomes of treatment actions, these changes need to be modeled.

  Other examples: tracking the location of a robot, tracking the economic activity of a nation, and making sense of a spoken or written sequence of words.

❖ States and observations

- Discussion on Discrete-time models, in which the world is viewed as a series of snapshots or time slices.
- Numbering the time slices 0, 1, 2, and so on, rather than assigning specific times to them
- Time interval Δ between slices is assumed to be the same for every interval this is dictated by the sensor; for example, a video camera might supply images at intervals of 1/30 of a second.
- In other cases, the interval is dictated by the typical rates of change of the relevant variables; for example, in the case of blood glucose monitoring, things can change significantly in the course of ten minutes, so a one-minute interval might be appropriate.
- On the other hand, in modeling continental drift over geological time, an interval of a million years might be fine.
- Uncertainty over continuous time can be modeled by stochastic differential equations (SDEs).
- Xt denotes the set of state variables at time t, which are assumed to
- be unobservable, and Et to denote the set of observable evidence variables.
- The observation at time t is Et = et for some set of values .

Consider the following example: 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 (whether the umbrella appears), and the set $\mathbf{X}_t$ contains a single state variable $Rain_t$ or R_t for short (whether it is raining).

In the diabetes example, the evidence variables might be $MeasuredBloodSugar_t$ and $PulseRate_t$ while the state variables might include $BloodSugar_t$ and $StomachContents_t$. (Notice that $BloodSugar_t$ and $MeasuredBloodSugar_t$ are not the same variable; this is how we deal with noisy measurements of actual quantities.)

state sequence starts at $t = 0$ and evidence starts arriving at $t = 1$. Hence, our umbrella world is represented by state variables $R_0, R_1, R_2, \ldots$ and evidence variables $U_1, U_2, \ldots$. 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 $\mathbf{X}_b$ inclusive. For example, $U_{1:3}$ corresponds to U_1, U_2, U_3.

- **Transition and Sensor Models**

- Next step is to specify how the world evolves (the transition model)
- How the evidence variables get their values (the sensor model).
- The transition model specifies the probability distribution over the latest state variables, given the previous values, that is,
- Problem: faced the set is unbounded in size as t increases.
- Problem by making a Markov assumption— that the current state depends on only a finite fixed number of previous states.
- Processes satisfying this assumption were first studied in depth by the statistician Andrei Markov (1856–1922) and are called Markov processes or Markov chains
- Current state depends only on the previous state and not on any earlier states.
- In other words, a state provides enough information to make the future conditionally independent of the past:

First order Markov process

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

Second Order Markov Process

$$\mathbf{P}(\mathbf{X}_t|\mathbf{X}_{t-2}, \mathbf{X}_{t-1}).$$

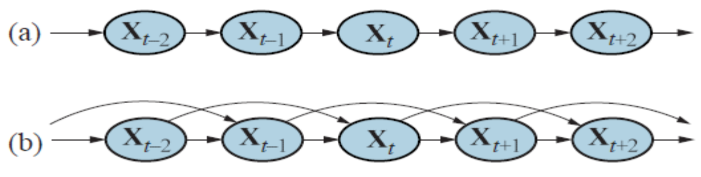


Figure 14.1 (a) Bayesian network structure corresponding to a first-order Markov process with state defined by the variables $\mathbf{X}_t$. (b) A second-order Markov process.

- Even with the Markov assumption there is still a problem: there are infinitely many possible values of t.
- This problem can be avoided by assuming that changes in the world state are caused by a time-homogeneous process- a process of change that is governed by laws that do not themselves change over time
- **Sensor Markov Assumption – Evidence variable depend on previous and current variable**

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

- The direction of the dependence between state and sensors: the arrows go from the actual state of the world to sensor values because the state of the world causes the sensors to take on particular values: the rain causes the umbrella to appear.
- The inference process, of course, goes in the other direction; the distinction between the direction of modeled dependencies and the direction of inference is one of the principal advantages of Bayesian networks.

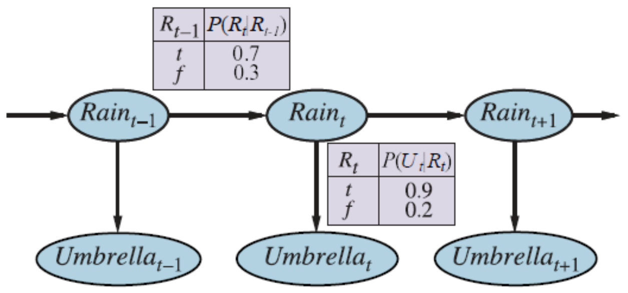


Figure 14.2 Bayesian network structure and conditional distributions describing the umbrella world. The transition model is $\mathbf{P}(Rain_t \mid Rain_{t-1})$ and the sensor model is $\mathbf{P}(Umbrella_t \mid Rain_t)$.

- In addition to specifying the transition and sensor models, we need to say how everything gets started—the prior probability distribution at time 0.

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

- The three terms on the right-hand side are the initial state model , the transition model , and the sensor model
- This equation defines the semantics of the family of temporal models represented by the three terms
- The first-order Markov assumption says that the state variables contain all the information needed to characterize the probability distribution for the next time slice.
- The assumption is only approximate, as in the case of predicting rain
- only on the basis of whether it rained the previous day.
- 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
  - Increasing the set of state variables. For example, we could add Season, Temperature, Humidity.

4.3.2 Inference in Temporal Models

- **Filtering and Prediction**
- **Smoothing**
- **Finding the most likely Sequence**

Having set up the structure of a generic temporal model, we can formulate the basic inference tasks that must be solved:

Filtering or state estimation- is the task of computing the belief state

$\mathbf{P}(\mathbf{X}_t|\mathbf{e}_{1:t})$-

- The posterior distribution over the most recent state given all evidence to date.
- In the umbrella example, this would mean computing the probability of rain today, given all the umbrella observations made so far

- Filtering is what a rational agent does to keep track of the current state so that rational decisions can be made

PREDICTION: This is the task of computing the posterior distribution over the future state, given all evidence to date.

- we Compute $\mathbf{P}(\mathbf{X}_{t+k}|\mathbf{e}_{1:t})$ for some k>0
- Prediction is useful for evaluating possible courses of action based on their expected outcomes.
- In the umbrella example, this might mean computing the probability of rain three days from now, given all the observations to date

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 for some k such that $\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t})$ 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 at time than was available at that time, because it incorporates more evidence

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 $\operatorname{argmax}_{\mathbf{x}_{1:t}} P(\mathbf{x}_{1:t}|\mathbf{e}_{1:t})$.
- 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

In addition to these inference tasks, we also have

LEARNING:

- The transition and sensor models, if not yet known, can be learned from
- observations.
- As with static Bayesian networks, dynamic Bayes net learning can be done as a by-product of inference.
- Inference provides an estimate of what transitions actually occurred and of what states generated the sensor readings, and these estimates can be used to learn the models.
- The learning process can operate via an iterative update algorithm called expectation–maximization or EM, or it can result from Bayesian updating of the model parameters given the evidence

❖ **Filtering and Prediction**

- 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
- Otherwise, the cost of each update increases as time goes by given the result of filtering up to time, the agent needs to compute the result for t + 1 from the new evidence et-1. So we have

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

Recursive estimation

- 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
  - This two-part process emerges quite simply when the formula is rearranged

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

α is a normalizing constant used to sum probabilities to 1.

  - The resulting equation for the new state estimate is:

$$\begin{aligned}\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1}) &= \alpha\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1})\sum_{\mathbf{x}_t}\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{x}_t, \mathbf{e}_{1:t})\,P(\mathbf{x}_t|\mathbf{e}_{1:t})\\ &= \alpha\,\underbrace{\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1})}_{\text{sensor model}}\sum_{\mathbf{x}_t}\underbrace{\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{x}_t)}_{\text{transition model}}\,\underbrace{P(\mathbf{x}_t|\mathbf{e}_{1:t})}_{\text{recursion}} \quad \text{(Markov assumption).}\end{aligned}$$

  - Filtered estimate $\mathbf{P}(\mathbf{X}_t|\mathbf{e}_{1:t})$ considered 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

$$\mathbf{f}_{1:t+1} = \text{FORWARD}(\mathbf{f}_{1:t}, \mathbf{e}_{t+1}),$$

process begins with $\mathbf{f}_{1:0} = \mathbf{P}(\mathbf{X}_0)$.

  - Where FORWARD implements the update described in Equation (14.5)
  - The time and space requirements for updating must be constant if a finite agent is to keep track of the current state distribution indefinitely
  - **Filtering and Prediction(Umbrella Example):**

Let us illustrate the filtering process for two steps in the basic umbrella example (Figure 14.2). That is, we will compute $\mathbf{P}(R_2|u_{1:2})$ 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 $U_1 = true$. The prediction from $t = 0$ to $t = 1$ is

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

Then the update step simply multiplies by the probability of the evidence for $t = 1$ and normalizes, as shown in Equation (14.4):

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

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

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

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

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

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

- 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

$$\mathbf{P}(\mathbf{X}_{t+k+1}|\mathbf{e}_{1:t}) = \sum_{\mathbf{x}_{t+k}} \underbrace{\mathbf{P}(\mathbf{X}_{t+k+1}|\mathbf{x}_{t+k})}_{\text{transition model}} \underbrace{P(\mathbf{x}_{t+k}|\mathbf{e}_{1:t})}_{\text{recursion}} .$$

- Naturally, this computation involves only the transition model and not the sensor model.
- The predicted distribution for rain converges to a fixed point , after which it remains constant for all time. This is the stationary distribution of the Markov process defined by the transition model.
- mixing time—roughly, the time taken to reach the fixed point.
- The more uncertainty there is in the transition model, the shorter will be the mixing time and the more the future is obscured.
- The forward recursion can be used to compute the likelihood of the evidence sequence
- message calculation is identical to that for filtering:

$$l_{1:t+1} = \text{FORWARD}(l_{1:t}, \mathbf{e}_{t+1}) .$$

Having computed $l_{1:t}$, we obtain the actual likelihood by summing out $\mathbf{X}_t$:

(14.7)

$$L_{1:t} = P(\mathbf{e}_{1:t}) = \sum_{\mathbf{x}_t} l_{1:t}(\mathbf{x}_t) .$$

❖ Smoothing

- As we said earlier, smoothing is the process of computing the distribution over past states given evidence up to the present:

  $\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t})$ for $0 \leq k < t$.

- we can split the computation into two parts—the evidence up to k and the evidence from k+1 to t,

(14.8)

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

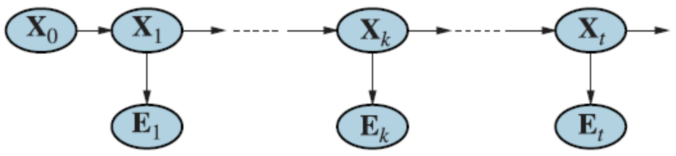


Figure 14.3 Smoothing computes $\mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:t})$, the posterior distribution of the state at some past time k given a complete sequence of observations from 1 to t.

where "$\times$" represents pointwise multiplication of vectors. Here we have defined a "backward" message $\mathbf{b}_{k+1:t} = \mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k)$, 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, as given by Equation (14.5). It turns out that the backward message $\mathbf{b}_{k+1:t}$ can be computed by a recursive process that runs *backward* from t:

(14.9)

$$\begin{aligned}\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k) &= \sum_{\mathbf{x}_{k+1}} \mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k,\mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k) \quad \text{(conditioning on } \mathbf{X}_{k+1})\\ &= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1:t}|\mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k) \quad \text{(by conditional independence)}\\ &= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1},\mathbf{e}_{k+2:t}|\mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k)\\ &= \sum_{\mathbf{x}_{k+1}} \underbrace{P(\mathbf{e}_{k+1}|\mathbf{x}_{k+1})}_{\text{sensor model}}\underbrace{P(\mathbf{e}_{k+2:t}|\mathbf{x}_{k+1})}_{\text{recursion}}\underbrace{\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k)}_{\text{transition model}},\end{aligned}$$

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

BACKWARD implements the update

For the initialization of the backward phase, we have $\mathbf{b}_{t+1:t} = \mathbf{P}(\mathbf{e}_{t+1:t}|\mathbf{X}_t) = \mathbf{P}(\ |\mathbf{X}_t) = \mathbf{1}$, where $\mathbf{1}$ is a vector of 1s. The reason for this is that $\mathbf{e}_{t+1:t}$ is an empty sequence, so the probability of observing it is 1.

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 (14.8), this is given by

(14.10)

$$\mathbf{P}(R_1|u_1, u_2) = \alpha\mathbf{P}(R_1|u_1)\mathbf{P}(u_2|R_1).$$

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 (14.9):

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

Plugging this into Equation (14.10), we find that the smoothed estimate for rain on day 1 is

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

Smoothed estimate for rain on day 1 is *higher than the filtered estimate (0.818) in*

this case.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.

```
function FORWARD-BACKWARD(ev, prior) returns a vector of probability dist
  inputs: ev, a vector of evidence values for steps 1, ..., t
          prior, the prior distribution on the initial state, P(X_0)
  local variables: fv, a vector of forward messages for steps 0, ..., t
                   b, a representation of the backward message, initially all 1s
                   sv, a vector of smoothed estimates for steps 1, ..., t

  fv[0] ← prior
  for i = 1 to t do
      fv[i] ← FORWARD(fv[i − 1], ev[i])
  for i = t down to 1 do
      sv[i] ← NORMALIZE(fv[i] × b)
      b ← BACKWARD(b, ev[i])
  return sv
```

Figure 14.4 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 (??) and (??), respectively.

❖ Finding the most likely Sequence”

- Suppose that [true, true, false, true, true] is the observed umbrella sequence for the security guard’s first five days on the job. What weather sequence is most likely to explain this?
- Does the absence of the umbrella on day 3 mean that it wasn’t raining,
  - did the director forget to bring it?
  - If it didn’t rain on day 3, perhaps (because weather tends to persist) it didn’t rain on day 4 either, but the director brought the umbrella just in case.
- In all, there are 25 possible weather sequences we could pick. Is there a way to find the most likely one, short of enumerating all of them and calculating their likelihoods?
- linear-time procedure can be used: 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
- There is a linear-time algorithm for finding the most likely sequence, but it requires more thought.
- It relies on the same Markov property that yielded efficient algorithms for filtering and smoothing.
- The idea is to 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 14.5(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.

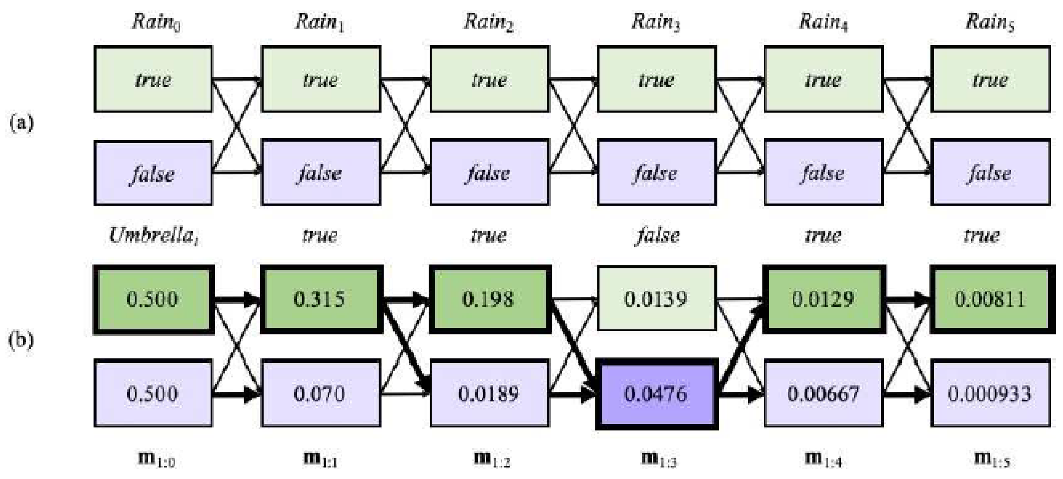


Figure 14.5 (a) Possible state sequences for $Rain_t$ can be viewed as paths through a graph of the possible states at each time step. (States are shown as rectangles to avoid confusion with nodes in a Bayes net.) (b) Operation of the Viterbi algorithm for the umbrella observation sequence $[true, true, false, true, true]$, where the evidence starts at time 1. 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, shown by the bold outlines and darker shading.

- We can use this property directly to construct a recursive algorithm for computing the most likely path given the evidence.
- We will use a recursively computed message, like the forward message in the filtering algorithm. The message is defined as:

$$\mathbf{m}_{1:t} = \max_{\mathbf{x}_{1:t-1}} \mathbf{P}\left(\mathbf{x}_{1:t-1}, \mathbf{X}_t, \mathbf{e}_{1:t}\right).$$

(14.11)

$$\begin{aligned}
\mathbf{m}_{1:t+1} &= \max_{\mathbf{x}_{1:t}} \mathbf{P}(\mathbf{x}_{1:t}, \mathbf{X}_{t+1}, \mathbf{e}_{1:t+1}) = \max_{\mathbf{x}_{1:t}} \mathbf{P}(\mathbf{x}_{1:t}, \mathbf{X}_{t+1}, \mathbf{e}_{1:t}, e_{t+1}) \\
&= \max_{\mathbf{x}_{1:t}} \mathbf{P}(e_{t+1}|\mathbf{x}_{1:t}, \mathbf{X}_{t+1}, \mathbf{e}_{1:t})\mathbf{P}(\mathbf{x}_{1:t}, \mathbf{X}_{t+1}, \mathbf{e}_{1:t}) \\
&= \mathbf{P}(e_{t+1}|\mathbf{X}_{t+1})\max_{\mathbf{x}_{1:t}} \mathbf{P}(\mathbf{X}_{t+1}, |\mathbf{x}_t)P(\mathbf{x}_{1:t}, \mathbf{e}_{1:t}) \\
&= \mathbf{P}(e_{t+1}|\mathbf{X}_{t+1})\max_{\mathbf{x}_t} \mathbf{P}(\mathbf{X}_{t+1}, |\mathbf{x}_t)\max_{\mathbf{x}_{1:t-1}} P(\mathbf{x}_{1:t-1}, \mathbf{x}_t, \mathbf{e}_{1:t})
\end{aligned}$$

4.4 Hidden Markov Model

4.4.1 Simplified Matrix Algorithms

4.4.2 Hidden Markov Model Example:Localization

- Hidden Markov model, or 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:Raint
- Although HMMs require the state to be a single, discrete variable, there is no corresponding restriction on the evidence variable

4.4.1 Simplified Matrix Algorithms

- With a single, discrete state variable, we can give concrete form to the representations of the transition model, the sensor model, and the forward and backward messages
- The state variable have values denoted by integers 1,…S , where S is the number of
- possible states.

The transition model $\mathbf{P}(X_t|X_{t-1})$ becomes an $S \times S$ matrix $\mathbf{T}$, where

$$\mathbf{T}_{ij} = P(X_t = j | X_{t-1} = i).$$

- That is,Tij is the probability of a transition from state to state . For example, if we number the states Rain=true and Rain=false as 1 and 2, respectively, then the transition matrix for the umbrella world defined as:

$$\mathbf{T} = \mathbf{P}(X_t|X_{t-1}) = \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix}.$$

- We can put sensor model in matrix form
- The value of the evidence variable Et is known at time t (call it), we need only specify, for each state, how likely it is that the state causes et to appear:
- we need $P(e_t|X_t = i)$ for each state i
- For mathematical convenience we place these values into an SxS diagonal observation matrix Ot
- if we use column vectors to represent the forward and backward messages, all the computations become simple matrix–vector operations.
- The forward equation

$$\mathbf{f}_{1:t+1} = \alpha \mathbf{O}_{t+1} \mathbf{T}^{\top} \mathbf{f}_{1:t}$$

- The backward equation

$$\mathbf{b}_{k+1:t} = \mathbf{T}\mathbf{O}_{k+1}\mathbf{b}_{k+2:t}.$$

- 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 requires the simultaneous presence of both the forward and backward messages
- For example, the "forward" message can be propagated backward if we manipulate Equation (14.12) to work in the other direction:

$$\mathbf{f}_{1:t} = \alpha'(\mathbf{f}^{\top})^{-1}\mathbf{O}_{t+1}^{-1}\mathbf{f}_{1:t+1}.$$

-
- A second area in which the matrix formulation reveals an improvement is in online
- smoothing with a fixed lag.
- The fact that smoothing can be done in constant space suggests that there should exist an efficient recursive algorithm for online smoothing—that is, an
- algorithm whose time complexity is independent of the length of the lag.
- Let us suppose that the lag is d; that is, we are smoothing at time slice t-d , where the current time is t .
- We compute

$$\alpha\mathbf{f}_{1:t-d} \times \mathbf{b}_{t-d+1:t}$$

(14.14)

$$\mathbf{b}_{t-d+1:t} = \left(\prod_{i=t-d+1}^{t} \mathbf{T}\mathbf{O}_i\right)\mathbf{b}_{t+1:t} = \mathbf{B}_{t-d+1:t}\mathbf{1},$$

where the matrix $\mathbf{B}_{t-d+1:t}$ is the product of the sequence of $\mathbf{T}$ and $\mathbf{O}$ matrices and $\mathbf{1}$ is a vector of 1s. $\mathbf{B}$ can be thought of as a "transformation operator" that transforms a later backward message into an earlier one. A similar equation holds for the new backward messages *after* the next observation arrives:

(14.15)

$$\mathbf{b}_{t-d+2:t+1} = \left(\prod_{i=t-d+2}^{t+1} \mathbf{T}\mathbf{O}_i\right)\mathbf{b}_{t+2:t+1} = \mathbf{B}_{t-d+2:t+1}\mathbf{1}.$$

```
function FIXED-LAG-SMOOTHING($e_t$, $hmm$, $d$) returns a distribution over $\mathbf{X}_{t-d}$
  inputs: $e_t$, the current evidence for time step $t$
          $hmm$, a hidden Markov model with $S \times S$ transition matrix $\mathbf{T}$
          $d$, the length of the lag for smoothing
  persistent: $t$, the current time, initially 1
              $\mathbf{f}$, the forward message $\mathbf{P}(X_t \mid e_{1:t})$, initially $hmm$.PRIOR
              $\mathbf{B}$, the $d$-step backward transformation matrix, initially the identity matrix
              $e_{t-d:t}$, double-ended list of evidence from $t-d$ to $t$, initially empty
  local variables: $\mathbf{O}_{t-d}, \mathbf{O}_t$, diagonal matrices containing the sensor model information

  add $e_t$ to the end of $e_{t-d:t}$
  $\mathbf{O}_t \leftarrow$ diagonal matrix containing $\mathbf{P}(e_t \mid X_t)$
  if $t > d$ then
     $\mathbf{f} \leftarrow$ FORWARD($\mathbf{f}, e_{t-d}$)
     remove $e_{t-d-1}$ from the beginning of $e_{t-d:t}$
     $\mathbf{O}_{t-d} \leftarrow$ diagonal matrix containing $\mathbf{P}(e_{t-d} \mid X_{t-d})$
     $\mathbf{B} \leftarrow \mathbf{O}_{t-d}^{-1}\mathbf{T}^{-1}\mathbf{B}\mathbf{T}\mathbf{O}_t$
  else $\mathbf{B} \leftarrow \mathbf{B}\mathbf{T}\mathbf{O}_t$
  $t \leftarrow t + 1$
  if $t > d + 1$ then return NORMALIZE($\mathbf{f} \times \mathbf{B1}$) else return null
```

Figure 14.6 An algorithm for smoothing with a fixed time lag of d steps, implemented as an online algorithm that outputs the new smoothed estimate given the observation for a new time step. Notice that the final output NORMALIZE($\mathbf{f} \times \mathbf{B1}$) is just $\alpha\, \mathbf{f} \times \mathbf{b}$, by Equation (??).

❖ Hidden Markov Model Example:Localization

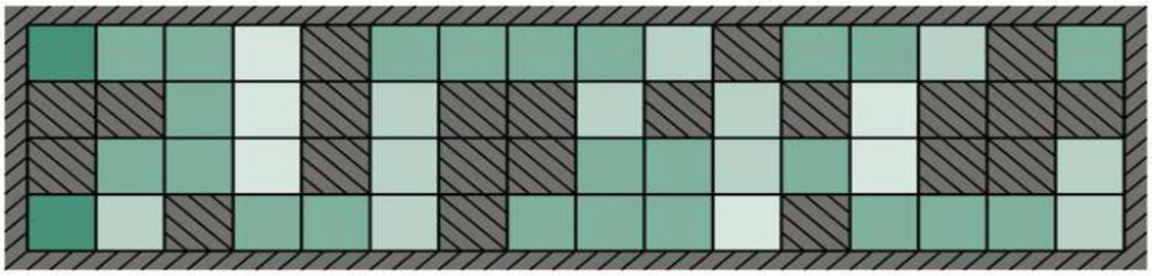
- localization problem for the vacuum world
- The robot has a single nondeterministic Move action and its sensors reported perfectly whether or not obstacles lay immediately to the north, south, east, and west
- The robot's belief state was the set of possible locations it could be in.
- We make the problem slightly more realistic by allowing for noise in the sensors, and formalizing the idea that the robot moves randomly—it is equally likely to move to any adjacent empty square.
- State variable Xt represents the location of the robot on the discrete grid; the domain of this variable is the set of empty squares – {1,… ,S}
- Let NEIGHBORS(i) be the set of empty squares that are adjacent to i
- N(i) be the size of that set.
- The transition model for the Move action says that the robot is equally likely to end up at any neighboring square

$$P(X_{t+1} = j | X_t = i) = \mathbf{T}_{ij} = \begin{cases} 1/N(i) \text{ if } j \in \text{NEIGHBORS}(i) \\ 0 \text{ otherwise.} \end{cases}$$

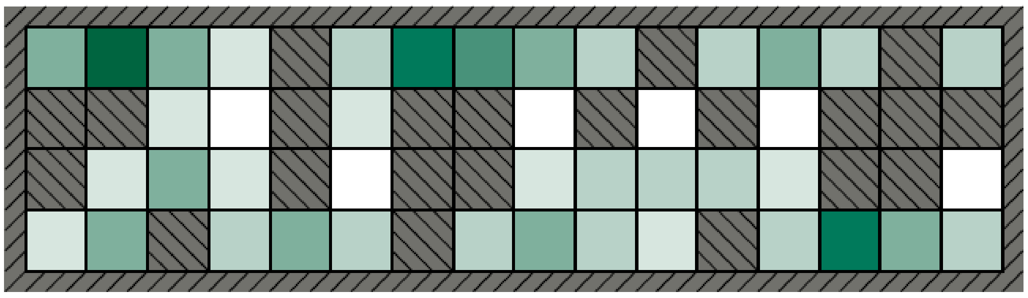
- We don't know where the robot starts, so we will assume a uniform distribution over all the squares;

$P(X_0 = i) = 1/S$. For the particular environment we consider (Figure 14.7), $S = 42$ and the transition matrix $\mathbf{T}$ has $42 \times 42 = 1764$ entries.

Figure 14.7



(a) Posterior distribution over robot location after $E_1 = 1011$



(b) Posterior distribution over robot location after $E_1 = 1011$, $E_2 = 1010$

Figure 14.7 Posterior distribution over robot location: (a) after one observation $E_1 = 1011$ (i.e., obstacles to the north, south, and west); (b) after a random move to an adjacent location and a second observation $E_2 = 1010$ (i.e., obstacles to the north and south). The size of each disk corresponds to the probability that the robot is at that location. The sensor error rate for each bit is $\epsilon = 0.2$.

- The sensor variable has 16 possible values, each a four-bit sequence giving the presence or absence of an obstacle in each of the compass directions NESW
- For example 1010, means that the north and south sensors report an obstacle and the east and west do not
- Each sensor's error rate is ϵ and that errors occur independently for the four sensor
- directions.
- Probability of getting all four bits right is: $(1 - \epsilon)^4$
- Probability of getting them all wrong is: ϵ^4.
- Discrepancy : dit
- 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—for example, where it began at time 0—and it can use the Viterbi algorithm to work out the most likely path it has taken to get where it is now

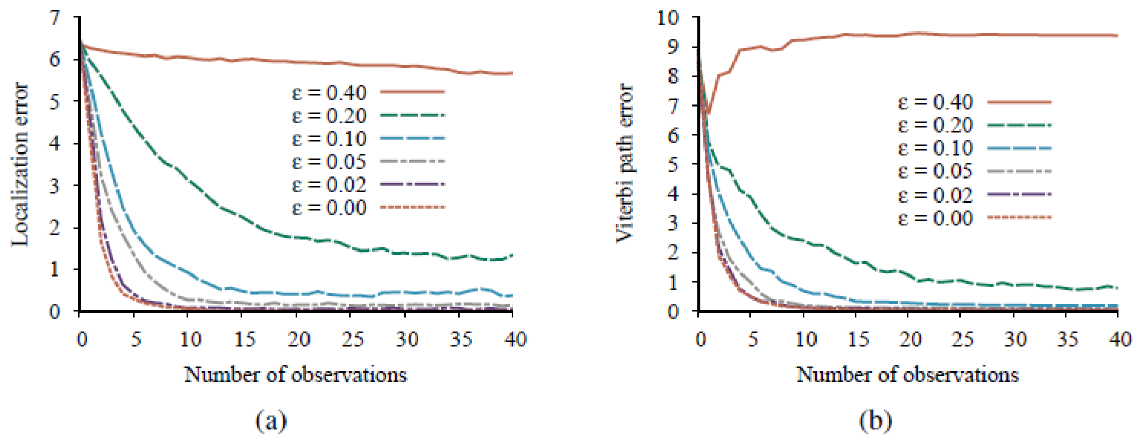


Figure 14.8 Performance of HMM localization as a function of the length of the observation sequence for various different values of the sensor error probability ϵ; data averaged over 400 runs. (a) The localization error, defined as the Manhattan distance from the true location. (b) The Viterbi path error, defined as the average Manhattan distance of states on the Viterbi path from corresponding states on the true path.

- HMMs have many uses in areas ranging from speech recognition to molecular biology
- They are fundamentally limited in their ability to represent complex processes.
- HMMs are an atomic representation: states of the world have no internal structure and are simply labeled by integers.

4.5 Kalman Filters

4.5.1 Updating Gaussian Distributions

4.5.2 A Simple One Dimensional Example

4.5.3 The general case

4.5.4 Applicability of 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.
- filtering: estimating state variables (here, the position and velocity of a moving object) from noisy observations over time.
- If the variables were discrete, we could model the system with a hidden Markov model
- To handle continuous variables, algorithm used is Kalman filtering, after one of its inventors, Rudolf Kalman.
- The bird's flight might be specified by six continuous variables at each time point; three for position (X_t, Y_t, Z_t)
- three for velocity $(\dot{X}_t, \dot{Y}_t, \dot{Z}_t)$
- Next state must be a linear function of the current state, plus some Gaussian noise

Let the time interval between observations be Δ, and assume constant velocity during the interval; then the position update is given by $X_{t+\Delta} = X_t + \dot{X}\Delta$. Adding Gaussian noise (to account for wind variation, etc.), we obtain a linear–Gaussian transition model:

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

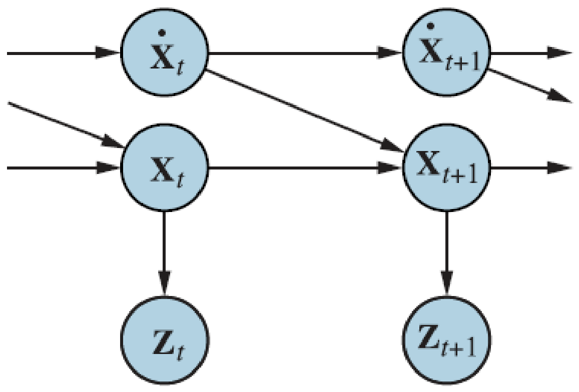


Figure 14.9 Bayesian network structure for a linear dynamical system with position $\mathbf{X}_t$, velocity $\dot{\mathbf{X}}_t$, and position measurement $\mathbf{Z}_t$.

4.5.1 Updating Gaussian Distributions

❖ **Two-step filtering calculation**

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

   (14.17)

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

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

   (14.18)

$$\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1}) = \alpha \mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1}) \mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t})$$

Thus, the FORWARD operator for Kalman filtering takes a Gaussian forward message $\mathbf{f}_{1:t}$, specified by a mean μ_t and covariance Σ_t, and produces a new multivariate Gaussian forward message $\mathbf{f}_{1:t+1}$, specified by a mean μ_{t+1} and covariance Σ_{t+1}. So if we start with a Gaussian prior $\mathbf{f}_{1:0} = \mathbf{P}(\mathbf{X}_0) = N(\mu_0, \Sigma_0)$, filtering with a linear–Gaussian model produces a Gaussian state distribution for all time.

4.5.2 A Simple One Dimensional Example

- FORWARD operator for the Kalman filter maps a Gaussian into a new
- Gaussian.
- This translates into computing a new mean and covariance from the previous
- mean and covariance.
- Deriving the update rule in the general (multivariate) case requires rather a lot of linear algebra
- The temporal model we consider describes a random walk of a single continuous state variable Xt with a noisy observation Zt

- An example might be the "consumer confidence" index, which can be modeled as undergoing a random Gaussian-distributed change each month measured by a random consumer survey that also introduces Gaussian sampling noise
- The prior distribution is assumed to be Gaussian with variance

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

transition model adds a Gaussian perturbation of constant variance σ_x^2 to the current state:

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

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

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

Now, given the prior $P(X_0)$, the one-step predicted distribution comes from Equation (14.17):

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

- The residual term can be taken outside the integral, giving us

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

- Plugging back in the expressions for a, b, and c and simplifying, we obtain
  Integral of gaussian over full range is 1

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

- To complete the update step, we need to condition on the observation at the first-time step, namely, Z1

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

- Once again, we combine the exponents and complete the square
  obtaining the following expression for the posterior:

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

- Thus, after one update cycle, we have a new Gaussian distribution for the state variable. New mean and Standard deviation:

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

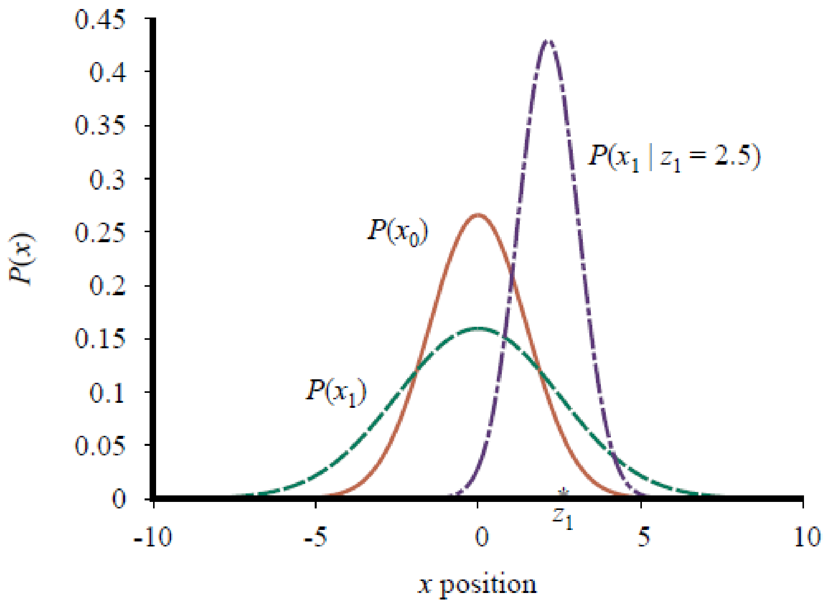


Figure 14.10 Stages in the Kalman filter update cycle for a random walk with a prior given by $\mu_0 = 0.0$ and $\sigma_0 = 1.5$, transition noise given by $\sigma_x = 2.0$, sensor noise given by $\sigma_z = 1.0$, and a first observation $z_1 = 2.5$ (marked on the x-axis). Notice how the prediction $P(x_1)$ 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.

4.5.3 General Case

- The full multivariate Gaussian distribution has the form

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

- Multiplying out the terms in the exponent, we see that the exponent is also a quadratic function of the values xi in x.
- Thus, filtering preserves the Gaussian nature of the state distribution.
- The general temporal model used with Kalman filtering.
- The transition model and the sensor model are required to be a linear transformation with additive Gaussian noise. Thus, we have

$$
\begin{aligned}
P(\mathbf{x}_{t+1}|\mathbf{x}_t) &= N(\mathbf{x}_{t+1};\mathbf{F}\mathbf{x}_t,\Sigma_x)\\
P(\mathbf{z}_t|\mathbf{x}_t) &= N(\mathbf{z}_t;\mathbf{H}\mathbf{x}_t,\Sigma_z),
\end{aligned}
$$

where $\mathbf{F}$ and Σ_x are matrices describing the linear transition model and transition noise covariance, and $\mathbf{H}$ and Σ_z are the corresponding matrices for the sensor model.

- Update equations for the mean and covariance

$$
\begin{aligned}
\mu_{t+1} &= \mathbf{F}\mu_t + \mathbf{K}_{t+1}(\mathbf{z}_{t+1} - \mathbf{H}\mathbf{F}\mu_t)\\
\Sigma_{t+1} &= (\mathbf{I} - \mathbf{K}_{t+1}\mathbf{H})(\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x),
\end{aligned}
$$

where $\mathbf{K}_{t+1} = (\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x)\mathbf{H}^\top(\mathbf{H}(\mathbf{F}\Sigma_t\mathbf{F}^\top + \Sigma_x)\mathbf{H}^\top + \Sigma_z)^{-1}$ is the **Kalman gain matrix.**

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 $\mathbf{X} = (X, Y, \dot{X}, \dot{Y})^\top$, so $\mathbf{F}$, Σ_x, $\mathbf{H}$, and Σ_z are 4×4 matrices.

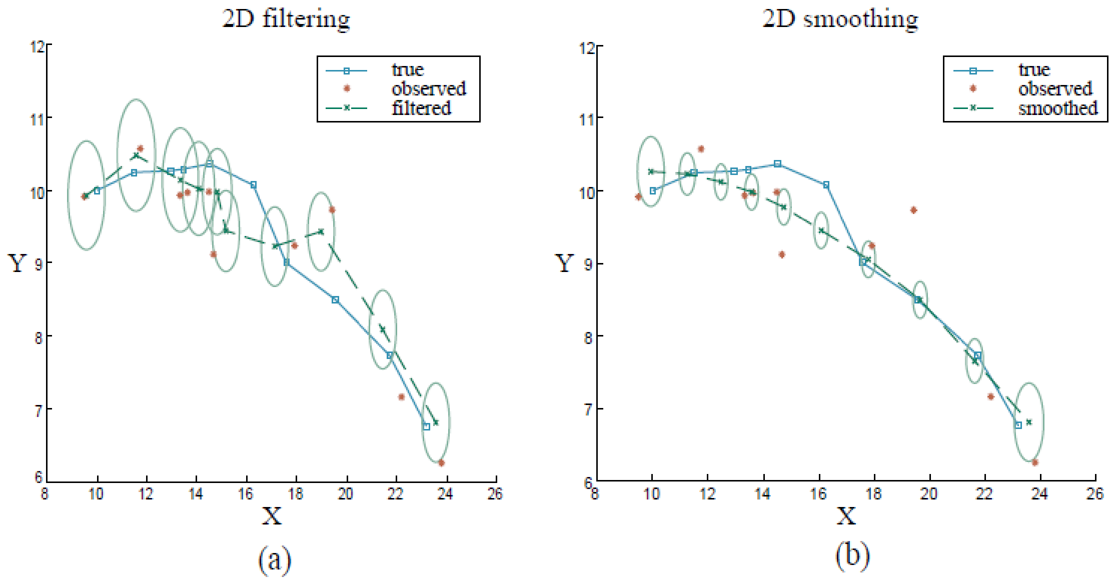


Figure 14.11 (a) Results of Kalman filtering for an object moving on the X–Y plane, showing the true trajectory (left to right), a series of noisy observations, and the trajectory estimated by Kalman filtering. Variance in the position estimate is indicated by the ovals. (b) The results of Kalman smoothing for the same observation sequence.

4.5.4 Applicability of Kalman Filtering

- The Kalman filter and its elaborations are used in a vast array of applications.
- 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.
- Kalman filters are used to reconstruct particle trajectories from bubble-chamber photographs and ocean currents from satellite surface measurements.
- 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.
- The extended Kalman filter (EKF) attempts to overcome nonlinearities in
- the system being modeled.
- A system is nonlinear if the transition model cannot be described as a matrix multiplication of the state vector

- This works well for smooth, well-behaved systems and allows the tracker to maintain and update a Gaussian state distribution that is a reasonable approximation to the true posterior

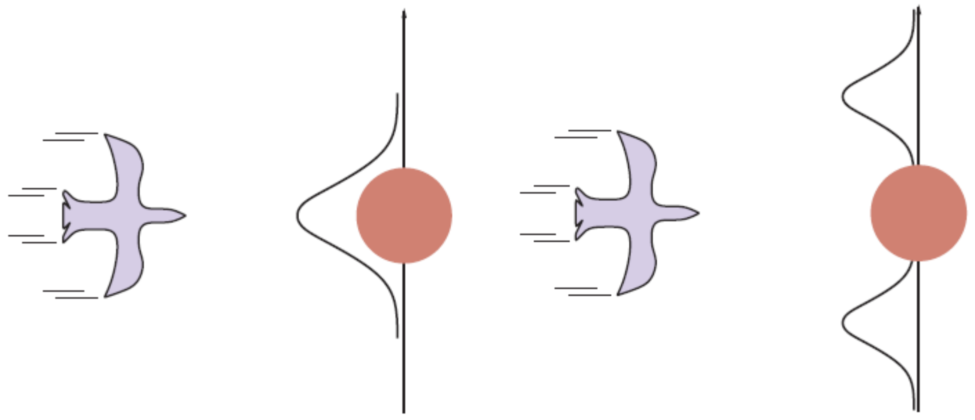


Figure 14.12 A bird flying toward a tree (top views). (a) A Kalman filter will predict the location of the bird using a single Gaussian centered on the obstacle. (b) A more realistic model allows for the bird's evasive action, predicting that it will fly to one side or the other.

❖ **Summary**

- The changing state of the world is handled by using a set of random variables to
- represent the state at each point in time.
- Representations can be designed to (roughly) satisfy the Markov property, so that the future is independent of the past given the present. Combined with the assumption that the process is time-homogeneous, this greatly simplifies the representation.
- A temporal probability model can be thought of as containing a transition model describing the state evolution and a sensor model describing the observation process.
- The principal inference tasks in temporal models are filtering (state estimation), prediction, smoothing, and computing the most likely explanation.
- Each of these tasks can be achieved using simple, recursive algorithms whose run time is linear in the length of the sequence.
- Two temporal models were studied in more depth: hidden Markov models, Kalman filter.