EXERCISES 14

PROBABILISTIC REASONING OVER TIME

14.1 Time and Uncertainty

Exercise 14.AUGM

Show that any second-order Markov process can be rewritten as a first-order Markov process with an augmented set of state variables. Can this always be done *parsimoniously*, i.e., without increasing the number of parameters needed to specify the transition model?

For each variable U_t that appears as a parent of a variable X_{t+2}, define an auxiliary variable U^{old}_{t+1}, such that U_t is parent of U^{old}_{t+1} and U^{old}_{t+1} is a parent of X_{t+2}. This gives us a first-order Markov model. To ensure that the joint distribution over the original variables is unchanged, we keep the CPT for X_{t+2} unchanged except for the new parent name, and we require that $\mathbf{P}(U^{old}_{t+1}|U_t)$ is an identity mapping, i.e., the child has the same value as the parent with probability 1. Since the parameters in this model are fixed and known, there is no effective increase in the number of free parameters in the model.

14.2 Inference in Temporal Models

Exercise 14.OBJT

Suppose that an object is moving according to the following transition model:

$$\begin{tikzcd}
A \arrow[loop above, "p"] \arrow[r, "1-p"] & B \arrow[loop above, "q"] \arrow[r, "1-q"] & C \arrow[loop above, "1"]
\end{tikzcd}$$

Here, $0 < p < 1$ and $0 < q < 1$ are arbitrary probabilities. At time 0, the object is known to be in state A.

a. What is the probability that the object is in A at time $n \geq 0$?

b. What is the probability that the object first reaches B at time $n \geq 1$?

c. What is the probability that the object is in B at time $n \geq 1$?

d. What is the probability that the object first reaches C at time $n \geq 2$?

e. What is the probability that the object is in C at time $n \geq 2$?

We define $\beta(n)$ and $\gamma(n)$ as the probability the object first reaches B and C, respectively at time n. We define $a(n), b(n), c(n)$ as the probability that the object is in A, B, C, respectively at time n.

a. For the object to be in A at time n, it must have stayed in A for n steps, which occurs with probability $a(n) \equiv p^n$.

b. For the object to first reach B at time n, it must have stayed in A for $n-1$ steps, then transitioned to B. This occurs with probability $\beta(n) \equiv a(n-1) \cdot (1-p) = p^{n-1}(1-p)$.

c. For the object to be in B at time n, it must have first reached B at time i for some $1 \leq i \leq n$, then stayed there for $n-i$ steps. Summing over all values of i gives

$$\begin{aligned} b(n) &\equiv \sum_{i=1}^{n} \beta(i) \cdot q^{n-i} \\ &= \sum_{i=1}^{n} p^{i-1}(1-p)q^{n-i} \\ &= \frac{(1-p)q^n}{p} \sum_{i=1}^{n} \left(\frac{p}{q}\right)^i \\ &= \frac{(1-p)q^n}{p} \cdot \frac{p}{q} \cdot \frac{1-\left(\frac{p}{q}\right)^n}{1-\frac{p}{q}} \\ &= (1-p)\frac{q^n - p^n}{q-p} \end{aligned}$$

where the simplifications in the last two lines assume $p \neq q$.

d. For the object to first reach C at time n, it must have been in B at time $n-1$, then transitioned to C. This occurs with probability

$$\gamma(n) \equiv b(n-1) \cdot (1-q) = (1-p)\frac{q^{n-1} - p^{n-1}}{q-p}(1-q) \, .$$

e. For the object to be in C at time n, it must not be in A or B at time n. This occurs with probability

$$1 - a(n) - b(n) = 1 - p^n - (1-p)\frac{q^n - p^n}{q-p} \, .$$

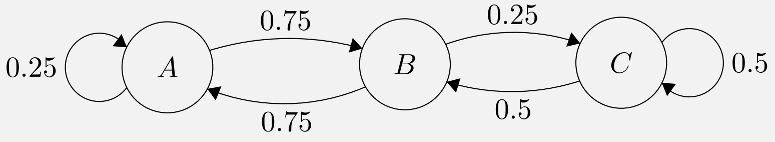
Alternatively, for the object to be in C at time n, it must have first reached C at time i for some $2 \leq i \leq n$, then stayed there for $n-i$ steps. Note that we can equivalently range over $1 \leq i \leq n$ for computational convenience, since $\gamma(1) = 0$. Summing over all values of i gives

$$\begin{aligned} c(n) &= \sum_{i=2}^{n} \gamma(i) \cdot 1^{n-i} \\ &= \sum_{i=1}^{n} \gamma(i) \cdot 1^{n-i} \\ &= \sum_{i=1}^{n} (1-p)\frac{q^{i-1}-p^{i-1}}{q-p}(1-q) \\ &= \frac{(1-p)(1-q)}{q-p}\left(\frac{1-q^n}{1-q} - \frac{1-p^n}{1-p}\right) \\ &= \frac{(1-p)(1-q^n)-(1-q)(1-p^n)}{q-p}, \end{aligned}$$

which is equivalent to the previous expression.

Exercise 14.STAD

Consider a Markov chain with 3 states and transition probabilities as shown below:



Compute the stationary distribution. That is, compute $P_\infty(A), P_\infty(B), P_\infty(C)$.

$P_\infty(A) = 0.4, P_\infty(B) = 0.4, P_\infty(C) = 0.2$

Exercise 14.CONV

In this exercise, we examine what happens to the probabilities in the umbrella world in the limit of long time sequences.

a. Suppose we observe an unending sequence of days on which the umbrella appears. Show that, as the days go by, the probability of rain on the current day increases monotonically toward a fixed point. Calculate this fixed point.

b. Now consider *forecasting* further and further into the future, given just the first two umbrella observations. First, compute the probability $P(r_{2+k}|u_1, u_2)$ for $k = 1 \ldots 20$ and plot the results. You should see that the probability converges towards a fixed point. Prove that the exact value of this fixed point is 0.5.

a. For all t, we have the filtering formula

$$\mathbf{P}(R_t|u_{1:t}) = \alpha \mathbf{P}(u_t|R_t) \sum_{R_{t-1}} \mathbf{P}(R_t|R_{t-1}) P(R_{t-1}|u_{1:t-1}) \,.$$

At the fixed point, we additionally expect that $\mathbf{P}(R_t|u_{1:t}) = \mathbf{P}(R_{t-1}|u_{1:t-1})$. Let the fixed-point probabilities be $\langle \rho,\ 1-\rho\rangle$. This provides us with a system of equations:

$$\begin{aligned}
\langle \rho,\ 1-\rho\rangle &= \alpha\langle 0.9,\ 0.2\rangle\langle 0.7,\ 0.3\rangle\rho + \langle 0.3,\ 0.7\rangle(1-\rho) \\
&= \alpha\langle 0.9,\ 0.2\rangle(\langle 0.4\rho,\ -0.4\rho\rangle + \langle 0.3,\ 0.7\rangle) \\
&= \frac{1}{0.9(0.4\rho+0.3)+0.2(-0.4\rho+0.7)}\langle 0.9,\ 0.2\rangle(\langle 0.4\rho,\ -0.4\rho\rangle + \langle 0.3,\ 0.7\rangle)
\end{aligned}$$

Solving this system, we find that $\rho \approx 0.8933$.

b. The probability converges to $\langle 0.5, 0.5\rangle$ as illustrated in Figure S14.1. This convergence makes sense if we consider a fixed-point equation for $\mathbf{P}(R_{2+k}|U_1, U_2)$:

$$\begin{aligned}
\mathbf{P}(R_{2+k}|U_1,U_2) &= \langle 0.7, 0.3\rangle P(r_{2+k-1}|U_1,U_2) + \langle 0.3, 0.7\rangle P(\neg r_{2+k-1}|U_1,U_2) \\
\mathbf{P}(r_{2+k}|U_1,U_2) &= 0.7P(r_{2+k-1}|U_1,U_2) + 0.3(1-P(r_{2+k-1}|U_1,U_2)) \\
&= 0.4P(r_{2+k-1}|U_1,U_2) + 0.3
\end{aligned}$$

That is, $P(r_{2+k}|U_1,U_2) = 0.5$.

Notice that the fixed point does not depend on the initial evidence.

Exercise 14.LIKL

Computing the evidence likelihood $L_{1:t} = P(\mathbf{e}_{1:t})$ in a temporal sequence can be done using a recursive computation similar to the filtering algorithm. Show that the **likelihood message** $\boldsymbol{\ell}_{1:t}(\mathbf{X}_t) = \mathbf{P}(\mathbf{X}_t, \mathbf{e}_{1:t})$ satisfies the same recursive relationship as the filtering message; that is,

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

$$\begin{aligned}
\boldsymbol{\ell}_{1:t+1} &= \mathbf{P}(\mathbf{X}_{t+1}, \mathbf{e}_{1:t+1}) \\
&= \mathbf{P}(\mathbf{X}_{t+1}, \mathbf{e}_{1:t}, \mathbf{e}_{t+1}) \\
&= \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}, \mathbf{e}_{1:t})\mathbf{P}(\mathbf{X}_{t+1}, \mathbf{e}_{1:t}) \\
&= \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1})\mathbf{P}(\mathbf{X}_{t+1}, \mathbf{e}_{1:t}) \\
&= \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1})\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t})\mathbf{P}(\mathbf{e}_{1:t}) \\
&= \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1})\sum_{x_t}\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t}, \mathbf{x}_t)\mathbf{P}(\mathbf{x}_t \mid \mathbf{e}_{1:t})\mathbf{P}(\mathbf{e}_{1:t}) \\
&= \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1})\sum_{x_t}\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t)\mathbf{P}(\mathbf{x}_t, \mathbf{e}_{1:t}) \\
&= \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1})\sum_{x_t}\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t)\boldsymbol{\ell}_{1:t}
\end{aligned}$$

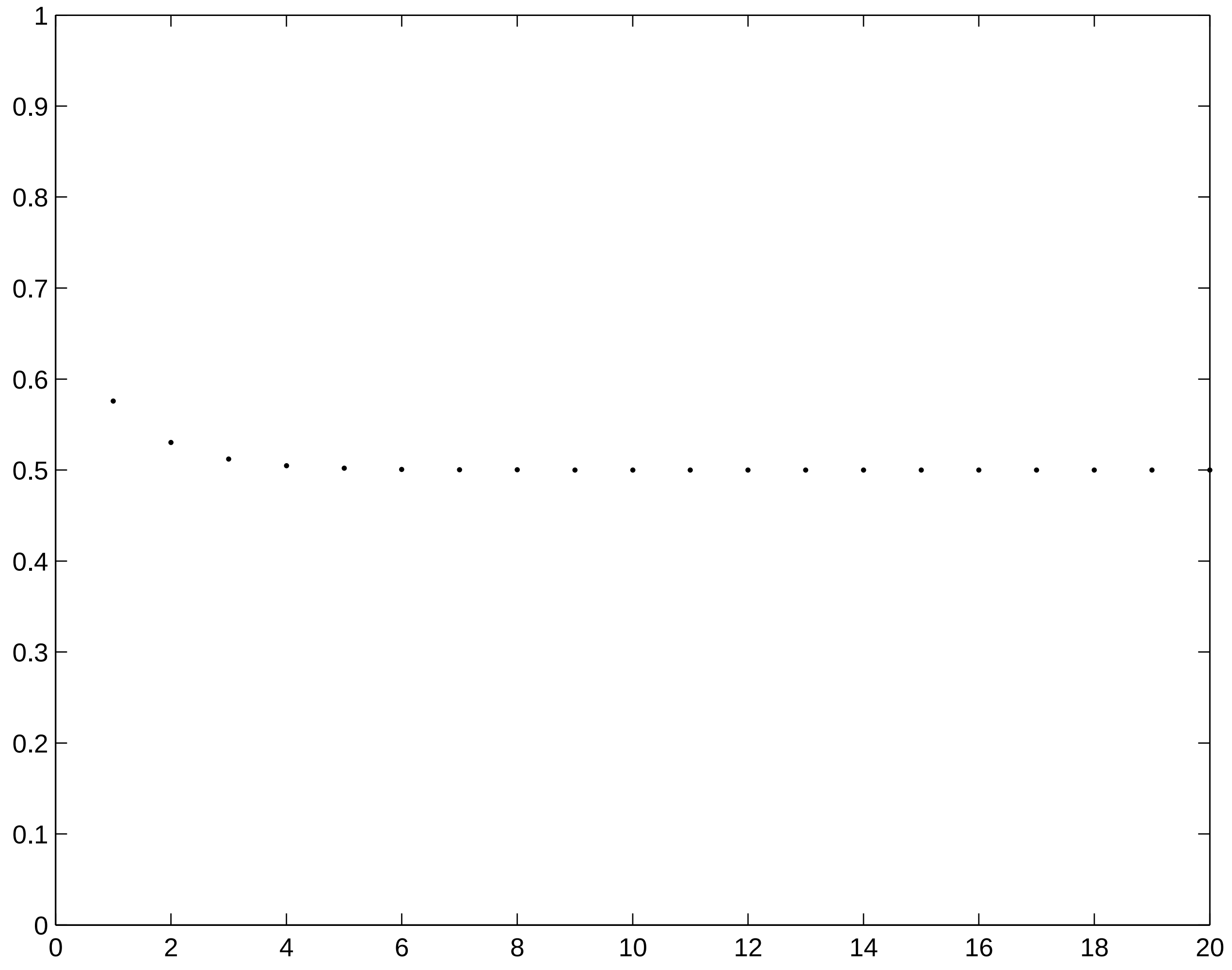


Figure S14.1 A graph of the probability of rain as a function of time, forecast into the future.

Exercise 14.ISLE

This exercise develops a space-efficient variant of the forward–backward algorithm described in Figure 14.4 (page 488). We wish to compute $\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t})$ for $k=1,\ldots,t$. This will be done with a divide-and-conquer approach.

a. Suppose, for simplicity, that t is odd, and let the halfway point be $h=(t+1)/2$. Show that $\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t})$ can be computed for $k=1,\ldots,h$ given just the initial forward message $\mathbf{f}_{1:0}$, the backward message $\mathbf{b}_{h+1:t}$, and the evidence $\mathbf{e}_{1:h}$.

b. Show a similar result for the second half of the sequence.

c. Given the results of (a) and (b), a recursive divide-and-conquer algorithm can be constructed by first running forward along the sequence and then backward from the end, storing just the required messages at the middle and the ends. Then the algorithm is called on each half. Write out the algorithm in detail.

d. Compute the time and space complexity of the algorithm as a function of t, the length of the sequence. How does this change if we divide the input into more than two pieces?

This exercise develops the Island algorithm for smoothing in DBNs (Binder *et al.*, 1997).

a. The chapter shows that $\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t})$ can be computed as

$$\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t}) = \alpha\,\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k})\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k) = \alpha\,\mathbf{f}_{1:k}\mathbf{b}_{k+1:t}$$

The forward recursion (Equation 15.3) shows that $\mathbf{f}_{1:k}$ can be computed from $\mathbf{f}_{1:k-1}$ and $\mathbf{e}_k$, which can in turn be computed from $\mathbf{f}_{1:k-2}$ and $\mathbf{e}_{k-1}$, and so on down to $\mathbf{f}_{1:0}$ and $\mathbf{e}_1$. Hence, $\mathbf{f}_{1:k}$ can be computed from $\mathbf{f}_{1:0}$ and $\mathbf{e}_{1:k}$. The backward recursion (Equation 15.7) shows that $\mathbf{b}_{k+1:t}$ can be computed from $\mathbf{b}_{k+2:t}$ and $\mathbf{e}_{k+1}$, which in turn can be computed from $\mathbf{b}_{k+3:t}$ and $\mathbf{e}_{k+2}$, and so on up to $\mathbf{b}_{h+1:t}$ and $\mathbf{e}_h$. Hence, $\mathbf{b}_{k+1:t}$ can be computed from $\mathbf{b}_{h+1:t}$ and $\mathbf{e}_{k+1:h}$. Combining these two, we find that $\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t})$ can be computed from $\mathbf{f}_{1:0}$, $\mathbf{b}_{h+1:t}$, and $\mathbf{e}_{1:h}$.

b. The reasoning for the second half is essentially identical: for k between h and t, $\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t})$ can be computed from $\mathbf{f}_{1:h}$, $\mathbf{b}_{t+1:t}$, and $\mathbf{e}_{h+1:t}$.

c. The algorithm takes 3 arguments: an evidence sequence, an initial forward message, and a final backward message. The forward message is propagated to the halfway point and the backward message is propagated backward. The algorithm then calls itself recursively on the two halves of the evidence sequence with the appropriate forward and backward messages. The base case is a sequence of length 1 or 2.

d. At each level of the recursion the algorithm traverses the entire sequence, doing $O(t)$ work. There are $O(\log_2 t)$ levels, so the total time is $O(t\log_2 t)$. The algorithm does a depth-first recursion, so the total space is proportional to the depth of the stack, i.e., $O(\log_2 t)$. With n islands, the recursion depth is $O(\log_n t)$, so the total time is $O(t\log_n t)$ but the space is $O(n\log_n t)$.

Exercise 14.VITE

On page 489, we outlined a flawed procedure for finding the most likely state sequence, given an observation sequence. The procedure involves finding the most likely state at each time step, using smoothing, and returning the sequence composed of these states. Show that, for some temporal probability models and observation sequences, this procedure returns an impossible state sequence (i.e., the posterior probability of the sequence is zero).

This is a very good exercise for deepening intuitions about temporal probabilistic reasoning. First, notice that the impossibility of the sequence of most likely states cannot come from an impossible observation because the smoothed probability at each time step includes the evidence likelihood at that time step as a factor. Hence, the impossibility of a sequence must arise from an impossible transition. Now consider such a transition from $X_k = i$ to $X_{k+1} = j$ for some i, j, k. For $X_{k+1} = j$ to be the most likely state at time $k+1$, even though it cannot be reached from the most likely state at time k, we can simply have an n-state system where, say, the smoothed probability of $X_k = i$ is $(1 + (n-1)\epsilon)/n$ and the remaining states have probability $(1-\epsilon)/n$. The remaining states all transition deterministically to $X_{k+1} = j$. From here, it is a simple matter to work out a specific model that behaves as desired.

14.3 Hidden Markov Models

Exercise 14.HMML

Equation (14.12) describes the filtering process for the matrix formulation of HMMs. Give a similar equation for the calculation of likelihoods, which was described generically in Equation (14.7).

The propagation of the $\boldsymbol{\ell}$ message is identical to that for filtering:

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

Since $\boldsymbol{\ell}$ is a column vector, each entry ℓ_i of which gives $P(X_t = i, \mathbf{e}_{1:t})$, the likelihood is obtained simply by summing the entries:

$$L_{1:t} = P(\mathbf{e}_{1:t}) = \sum_i \ell_i \,.$$

Exercise 14.VACS

Consider the vacuum worlds of Figure 4.18 (perfect sensing) and Figure 14.7 (noisy sensing). Suppose that the robot receives an observation sequence such that, with perfect sensing, there is exactly one possible location it could be in. Is this location necessarily the most probable location under noisy sensing for sufficiently small noise probability ϵ? Prove your claim or find a counterexample.

Let ℓ be the single possible location under deterministic sensing. Certainly, as $\epsilon \to 0$, we expect intuitively that $P(X_t = \ell \mid e_{1:t}) \to 1$. If we assume that all reachable locations are equally likely to be reached under the uniform motion model, then the claim that ℓ is the most likely location under noisy sensing follows immediately: any other location must entail at least one sensor discrepancy—and hence a probability penalty factor of ϵ—on every path reaching it in $t-1$ steps, otherwise it would be logically possible in the deterministic setting. The assumption is incorrect, however: if the neighborhood graph has outdegree k, the probability of reaching any two locations could differ by a factor of $O(k^t)$. If we set ϵ smaller than this, the claim still holds. But for any fixed ϵ, there are neighborhood graphs and observation sequences such that the claim may be false for sufficiently large t. Essentially, if $t-1$ steps of random movement are much more likely to reach m than ℓ—e.g., if ℓ is at

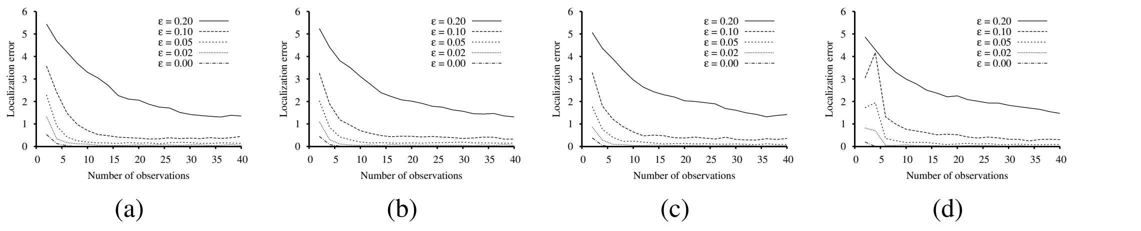


Figure S14.2 Graphs showing the expected localization error as a function of time, for bias values of 1.0 (unbiased), 2.0, 5.0, 10.0.

the end of a long tunnel of length exactly $t-1$—then that can outweigh the cost of a sensor error or two. Notice that this argument requires an environment of unbounded size; for any bounded environment, we can bound the reachability ratio and set ϵ accordingly.

Exercise 14.HMMR

In Section 14.3.2, the prior distribution over locations is uniform and the transition model assumes an equal probability of moving to any neighboring square. What if those assumptions are wrong? Suppose that the initial location is actually chosen uniformly from the northwest quadrant of the room and the *Move* action actually tends to move southeast. Keeping the HMM model fixed, explore the effect on localization and path accuracy as the southeasterly tendency increases, for different values of ϵ.

This exercise is an important one: it makes very clear the difference between the actual environment and the agent's model of it. To generate the required data, the student will need to run a world simulator (movement and percepts) using the true model (northwest prior, southeast movement tendency), while running the agent's state estimator using the assumed model (uniform prior, uniformly random movement). The student will also begin to appreciate the inexpressiveness of HMMs after constructing the 64×64 transition matrix from the more natural representation in terms of coordinates.

Perhaps surprisingly, the data for expected localization error (expected Manhattan distance between true location and the posterior state estimate) show that having an incorrect model is not too problematic. A "southeast" bias of b was implemented by multiplying the probability of any south or east move by b and then renormalizing the distribution before sampling. Graphs for four different values of the bias are shown in Figure S14.2. The results suggest that the sensor data sequence overwhelms any error introduced by the incorrect motion model.

Exercise 14.ROOM

Consider a version of the vacuum robot (page 495) that has the policy of going straight for as long as it can; only when it encounters an obstacle does it change to a new (randomly selected) heading. To model this robot, each state in the model consists of a *(location, head-*

ing) pair. Implement this model and see how well the Viterbi algorithm can track a robot with this model. The robot's policy is more constrained than the random-walk robot; does that mean that predictions of the most likely path are more accurate?

The code for this exercise is very similar to that for Exercise 14.HMMR. The main difference is the state space: instead of 64 locations, the state space has 256 location–heading pairs, and the transition matrix is 256×256—starting to be a little painful when running hundreds of trials. We also need to add a "bump" bit to the percept vector, and we assume this is perfectly observed (else the robot could not always decide to pick a new heading). Generally we expect localization to be more accurate, since the sensor sequence need only disambiguate among a small number of possible headings rather than an exponentially growing set of random-walk paths. Also the exact bump sensor will eliminate many possible states completely.

Exercise 14.FILL

This exercise is concerned with filtering in an environment with no landmarks. Consider a vacuum robot in an empty room, represented by an $n \times m$ rectangular grid. The robot's location is hidden; the only evidence available to the observer is a noisy location sensor that gives an approximation to the robot's location. If the robot is at location (x, y) then with probability .1 the sensor gives the correct location, with probability .05 each it reports one of the 8 locations immediately surrounding (x, y), with probability .025 each it reports one of the 16 locations that surround those 8, and with the remaining probability of .1 it reports "no reading." The robot's policy is to pick a direction and follow it with probability .8 on each step; the robot switches to a randomly selected new heading with probability .2 (or with probability 1 if it encounters a wall). Implement this as an HMM and do filtering to track the robot. How accurately can we track the robot's path?

The code for this exercise is very similar to that for Exercise 14.ROOM. The state is again a location/heading pair with a 256×256 transition matrix. The observation model is different: instead of a 4-bit percept (16 possible percepts), the percept is a location (or null), for $n \times m + 1$ possible percepts. Generally speaking, tracking works well near the walls because any bump (or even the lack thereof) helps to pin down the location. Away from the walls, the location uncertainty will be slightly worse but still generally accurate because the location errors are independent and unbiased and the policy is fairly predictable. It is reasonable to expect students to provide snapshots or movies of true location and posterior estimate, particularly if they are given suitable graphics infrastructure to make this easy.

Exercise 14.HMME

Consider an HMM with state variables $\{X_i\}$ and emission variables $\{Y_i\}$.

(i) [*true* or *false*] X_i is always conditionally independent of Y_{i+1} given X_{i+1}.

(ii) [*true* or *false*] There exists an HMM where X_i is conditionally independent of Y_i given X_{i+1}.

(iii) [*true* or *false*] If $Y_i = X_i$ with probability 1, and the state space is of size k, then the most efficient algorithm for computing $p(X_t|y_1 \cdots, y_t)$ takes $O(k)$ or less time.

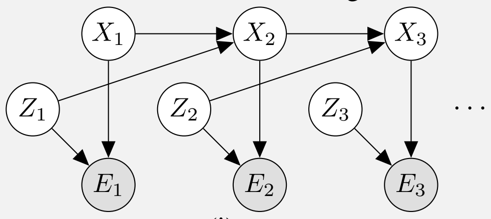
True, True, True

Exercise 14.ELPS

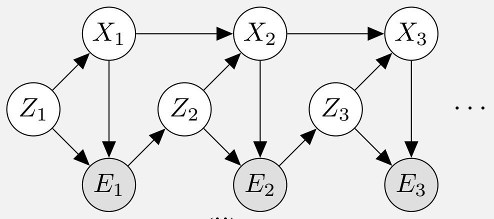
Recall that for a standard HMM the Elapse Time update and the Observation update are of the respective forms:

$$P(X_t \mid e_{1:t-1}) = \textstyle\sum_{x_{t-1}} P(X_t \mid x_{t-1}) P(x_{t-1} \mid e_{1:t-1})$$
$$P(X_t \mid e_{1:t}) \propto P(X_t \mid e_{1:t-1}) P(e_t \mid x_t)$$

We now consider the following two HMM-like models:



(i)



(ii)

Derive the modified Elapse Time update and the modified Observation update that correctly compute the beliefs from the quantities that are available in the Bayes' Net.

(i) First Bayes Net

(a) In the elapse time update, we want to get from $P(X_{t-1}, Z_{t-1}|e_{1:t-1})$ to $P(X_t, Z_t|e_{1:t-1})$.

$$\begin{aligned} P(X_t, Z_t|e_{1:t-1}) &= \sum_{x_{t-1}, z_{t-1}} P(X_t, Z_t, x_{t-1}, z_{t-1}|e_{1:t-1}) \\ &= \sum_{x_{t-1}, z_{t-1}} P(x_{t-1}, z_{t-1}|e_{1:t-1}) P(X_t|x_{t-1}, z_{t-1}, e_{1:t-1}) \\ &\quad P(Z_t|X_t, x_{t-1}, z_{t-1}, e_{1:t-1}) \\ &= \sum_{x_{t-1}, z_{t-1}} P(x_{t-1}, z_{t-1}|e_{1:t-1}) P(X_t|x_{t-1}, z_{t-1}) P(Z_t) \end{aligned}$$

First line: marginalization, second line: chain rule, third line: conditional independence assumptions.

(b) In the observation update, we want to get from $P(X_t, Z_t|e_{1:t-1})$ to $P(X_t, Z_t|e_{1:t})$.

$$\begin{aligned} P(X_t, Z_t|e_{1:t}) &\propto P(X_t, Z_t, e_t|e_{1:t-1}) \\ &\propto P(X_t, Z_t|e_{1:t-1}) P(e_t|X_t, Z_t, e_{1:t-1}) \\ &\propto P(X_t, Z_t|e_{1:t-1}) P(e_t|X_t, Z_t) \end{aligned}$$

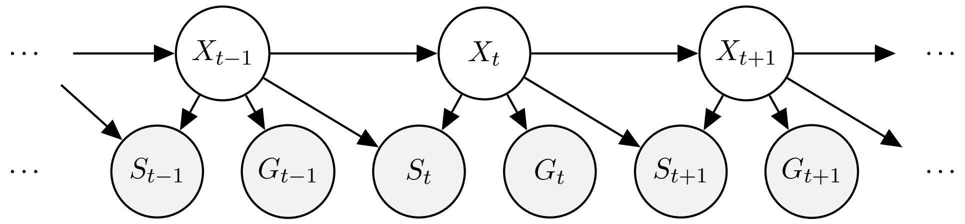


Figure S14.3 HMM for Exercise 14.TRFC.

First line: normalization, second line: chain rule, third line: conditional independence assumptions.

(ii) Second Bayes Net

(a) In the elapse time update, we want to get from $P(X_{t-1}, Z_{t-1}|e_{1:t-1})$ to $P(X_t, Z_t|e_{1:t-1})$.

$$\begin{aligned}
&P(X_t, Z_t|e_{1:t-1}) \\
&= \sum_{x_{t-1},z_{t-1}} P(X_t, Z_t, x_{t-1}, z_{t-1}|e_{1:t-1}) \\
&= \sum_{x_{t-1},z_{t-1}} P(x_{t-1}, z_{t-1}|e_{1:t-1})P(Z_t|x_{t-1}, z_{t-1}, e_{1:t-1})P(X_t|Z_t, x_{t-1}, z_{t-1}, e_{1:t-1}) \\
&= \sum_{x_{t-1},z_{t-1}} P(x_{t-1}, z_{t-1}|e_{1:t-1})P(Z_t|e_{t-1})P(X_t|x_{t-1}, Z_t)
\end{aligned}$$

First line: marginalization, second line: chain rule, third line: conditional independence assumptions.

(b) In the observation update, we want to get from $P(X_t, Z_t|e_{1:t-1})$ to $P(X_t, Z_t|e_{1:t})$.

$$\begin{aligned}
P(X_t, Z_t|e_{1:t}) &\propto P(X_t, Z_t, e_t|e_{1:t-1}) \\
&\propto P(X_t, Z_t|e_{1:t-1})P(e_t|X_t, Z_t, e_{1:t-1}) \\
&\propto P(X_t, Z_t|e_{1:t-1})P(e_t|X_t, Z_t)
\end{aligned}$$

First line: normalization, second line: chain rule, third line: conditional independence assumptions.

Exercise 14.TRFC

Transportation researchers are trying to improve traffic in the city but, in order to do that, they first need to estimate the location of each of the cars in the city. They need our help to model this problem as an inference problem of an HMM. For this question, assume that only *one* car is being modeled.

a. We define the variables as follows: X, the location of the car; S, the noisy location of the car from the signal strength at a nearby cell phone tower; and G, the noisy location of the car from GPS. In our HMM model Figure S14.3, the signal-dependent location S_t not only depends on the current state X_t but also depends on the previous state X_{t-1}.

We want to compute the belief $P(x_t|s_{1:t}, g_{1:t})$. In this part we consider an update that combines the dynamics and observation update in a *single* update. Complete the **forward update** expression by filling in the expression of the form:

$$P(x_t|s_{1:t}, g_{1:t}) = \underline{\hspace{8em}} \quad P(x_{t-1}|s_{1:t-1}, g_{1:t-1}).$$

b. We consider extending the Viterbi algorithm for the HMM from the previous part. We want to find the most likely sequence of states $X_{1:T}$ given the sequence of observations $s_{1:T}$ and $g_{1:T}$. Find the dynamic programming update for $t > 1$ for the modified HMM of the following form:

$$m_t[x_t] = \underline{\hspace{7em}} \quad m_{t-1}[x_{t-1}].$$

a. For this modified HMM, we have the dynamics and observation update in a single update because one of the previous independence assumptions does not longer holds.

$$\begin{aligned}
P(x_t|s_{1:t}, g_{1:t}) &= \sum_{x_{t-1}} P(x_{t-1}, x_t|s_t, g_t, s_{1:t-1}, g_{1:t-1}) \\
&= \frac{1}{P(s_t, g_t|s_{1:t-1}, g_{1:t-1})} \sum_{x_{t-1}} P(x_{t-1}, x_t, s_t, g_t|s_{1:t-1}, g_{1:t-1}) \\
&= \frac{1}{P(s_t, g_t|s_{1:t-1}, g_{1:t-1})} \sum_{x_{t-1}} P(s_t, g_t|x_{t-1}, x_t, s_{1:t-1}, g_{1:t-1}) \\
&\quad P(x_{t-1}, x_t|s_{1:t-1}, g_{1:t-1}) \\
&= \frac{1}{P(s_t, g_t|s_{1:t-1}, g_{1:t-1})} \sum_{x_{t-1}} P(s_t, g_t|x_{t-1}, x_t) P(x_t|x_{t-1}, s_{1:t-1}, g_{1:t-1}) \\
&\quad P(x_{t-1}|s_{1:t-1}, g_{1:t-1}) \\
&= \frac{1}{P(s_t, g_t|s_{1:t-1}, g_{1:t-1})} \sum_{x_{t-1}} P(s_t|x_{t-1}, x_t) P(g_t|x_{t-1}, x_t) P(x_t|x_{t-1}) \\
&\quad P(x_{t-1}|s_{1:t-1}, g_{1:t-1}) \\
&= \frac{1}{P(s_t, g_t|s_{1:t-1}, g_{1:t-1})} \sum_{x_{t-1}} P(s_t|x_{t-1}, x_t) P(g_t|x_t) P(x_t|x_{t-1}) \\
&\quad P(x_{t-1}|s_{1:t-1}, g_{1:t-1})
\end{aligned}$$

In the third to last step, we use the independence assumption $S_t, G_t \perp\!\!\!\perp S_{1:t-1}, G_{1:t-1}|X_{t-1}, X_t$; in the second to last step, we use the independence assumption $S_t \perp\!\!\!\perp G_t|X_{t-1}, X_t$ and $X_t \perp\!\!\!\perp S_{1:t-1}, G_{1:t-1}|X_{t-1}$; and in the last step, we use the independence assumption

$G_t \perp\!\!\!\perp X_{t-1}|X_t$.

b. If we remove the summation from the forward update equation of part **(b)**, we get a joint probability of the states,

$$P(x_{1:t}|s_{1:t}, g_{1:t}) = \frac{P(s_t|x_{t-1}, x_t)P(g_t|x_t)P(x_t|x_{t-1})}{P(s_t, g_t|s_{1:t-1}, g_{1:t-1})} P(x_{1:t-1}|s_{1:t-1}, g_{1:t-1}).$$

We can define $m_t[x_t]$ to be the maximum joint probability of the states (for a particular x_t) given all past and current observations, times some constant, and then we can find a recursive relationship for $m_t[x_t]$,

$$\begin{aligned}
m_t[x_t] &= P(s_{1:t}, g_{1:t}) \max_{x_{1:t-1}} P(x_{1:t}|s_{1:t}, g_{1:t}) \\
&= P(s_{1:t}, g_{1:t}) \max_{x_{1:t-1}} \frac{P(s_t|x_{t-1}, x_t)P(g_t|x_t)P(x_t|x_{t-1})}{P(s_t, g_t|s_{1:t-1}, g_{1:t-1})} P(x_{1:t-1}|s_{1:t-1}, g_{1:t-1}) \\
&= \max_{x_{t-1}} P(s_t|x_{t-1}, x_t)P(g_t|x_t)P(x_t|x_{t-1}) \frac{P(s_{1:t}, g_{1:t})}{P(s_t, g_t|s_{1:t-1}, g_{1:t-1})} \\
&\quad \max_{x_{1:t-2}} P(x_{1:t-1}|s_{1:t-1}, g_{1:t-1}) \\
&= \max_{x_{t-1}} P(s_t|x_{t-1}, x_t)P(g_t|x_t)P(x_t|x_{t-1})P(s_{1:t-1}, g_{1:t-1}) \\
&\quad \max_{x_{1:t-2}} P(x_{1:t-1}|s_{1:t-1}, g_{1:t-1}) \\
&= \max_{x_{t-1}} P(s_t|x_{t-1}, x_t)P(g_t|x_t)P(x_t|x_{t-1})m_{t-1}[x_{t-1}].
\end{aligned}$$

Notice that the maximum joint probability of states up to time $t = T$ given all past and current observations is given by

$$\max_{x_{1:T}} P(x_{1:T}|s_{1:T}, g_{1:T}) = \frac{\max_{x_t} m_T[x_t]}{P(s_{1:T}, g_{1:T})}.$$

We can recover the actual most likely sequence of states by bookkepping back pointers of the states the maximized the Viterbi update equations.

Exercise 14.ELEV

Assume the elevator of the Disney Tower of Terror E follows a Markovian process and has m floors at which it can stop. In the dead of night, you install a sensor S at the top of the shaft that gives approximate distance measurements, quantized into n different distance bins. Assume that the elevator stops at T floors as part of the ride and the initial distribution of the elevator is uniform over the m floors as shown in Figure S14.4.

You want to know the most probable sequence of (hidden) states $X_{1:T}$ given your observations $y_{1:T}$ from the sensor, so you turn to the Viterbi algorithm, which performs the following update at each step:

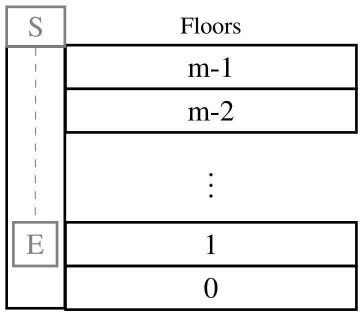


Figure S14.4 Disney Tower for Exercise 14.ELEV.

$$m_t[x_t] = P(y_t|x_t) \max_{x_{t-1}} [P(x_t|x_{t-1}) m_{t-1}[x_{t-1}]]$$

$$a_t[x_t] = \arg\max_{x_{t-1}} m_{t-1}[x_{t-1}]$$

a. What is the runtime and space complexity of the Viterbi algorithm to determine all previous states for this scenario?

b. If we only want to determine the previous K states, what is the runtime and space complexity of the Viterbi algorithm to determine the previous K states?

c. Suppose you instead only wish to determine the current distribution (at time T) for the elevator, given your T observations, so you use the forward algorithm, with update step shown here:

$$P(X_t|y_t) \propto P(y_t|x_t) \sum_{x_{t-1}} P(X_t|x_{t-1}) P(x_{t-1}, y_{0:t-1})$$

Additionally, from your previous analysis, you note that there are some states which are unreachable from others (e.g., the elevator cannot travel from the top floor to the bottom in a single timestep). Specifically, from each state, there are between $G/2$ and G states which can be reached in the next timestep, where $G < m$. What is the runtime and space complexity for the forward algorithm to estimate the current state at time T, assuming we ignore states that cannot be reached in each update?

d. Suppose you decide to use a particle filter instead of the forward algorithm. What is the runtime of a particle filter with P particles?

a. **Runtime.** For each timestep that we want to decode, we loop over all states and consider the transition probability to that state from all previous states. Thus, the runtime is Tm^2, since we repeat this process for each timestep (T), and at each timestep we have a double for loop for states (one to compute m-value for all states, one to compute arg max of previous m-values and transition probabilities, so m^2).

Space Complexity. We store only our a-value arrays (which keep track of the best transition along the path to the current state). These are essentially back-pointers so we can reconstruct the best path. For a sequence of length T, we then have a matrix of these pointers that is $T \times m$, so we take up Tm space.

b. **Runtime.** The runtime is the same as part a since the forward pass part of the Viterbi algorithm is unchanged (e.g., we still need to consider all transition probabilities at all timesteps to determine the most likely path, regardless of how far back we want to go).

Space Complexity. If we only wish to determine the previous K states, we only need to store a-values corresponding to the previous K timesteps (since when we reconstruct the path, we only need to go K steps back). Thus, the space complexity decreases to Km from part a.

c. **Runtime.** The normal runtime for the forward algorithm is Tm^2 (very similar to Viterbi - we consider transition probabilities from each state to each other state during our forward pass). However, if we can ignore those states with zero transition probability, then we only need to consider G states when calculating our sum for each state. Thus, the runtime is reduced to TmG.

Space Complexity. The space complexity for the forward algorithm is unchanged by how many states we can reach from each state, so this is the same as the normal space complexity for the forward algorithm. It is m because we store $P(X_t|y_t)$ for each state and we only store one timestep (since we only care about the current state probabilities).

d. Assuming the time for resampling a single particle is constant, then the runtime is TP. At each timestep (total of T), we perform the time elapse and observation updates in constant time for each of the P particles.

14.4 Particle Filtering

Exercise 14.PFTW

Consider two particle filtering implementations:

Implementation 1: Initialize particles by sampling from initial state distribution and assigning uniform weights.

1. Propagate particles, retaining weights
2. Resample according to weights
3. Weight according to observations

Implementation 2: Initialize particles by sampling from initial state distribution.

1. Propagate unweighted particles
2. Weight according to observations
3. Resample according to weights

(i) [*true* or *false*] Implementation 2 will typically provide a better approximation of the estimated distribution than implementation 1.

(ii) [*true* or *false*] If the transition model is deterministic then both implementations provide equally good estimates of the distribution.

(iii) [*true* or *false*] If the observation model is deterministic then both implementations provide equally good estimates of the distribution.

True, True, False

Exercise 14.PFCD

(iv) [*true* or *false*] With a deterministic transition model and a stochastic observation model, as time goes to infinity, when running a particle filter we will end up with all identical particles.

(v) [*true* or *false*] With a deterministic observation model, all particles might end up having zero weight.

(vi) [*true* or *false*] It is possible to use particle filtering when the state space is continuous.

(vii) [*true* or *false*] It is possible to use particle filtering when the state space is discrete.

(viii) [*true* or *false*] As the number of particles goes to infinity, particle filtering will represent the same probability distribution that you'd get by using exact inference.

(ix) [*true* or *false*] Particle filtering can represent a flat distribution (i.e. uniform) with fewer particles than it would need for a more concentrated distribution (i.e. Gaussian).

True, True, True, True, True, False

Exercise 14.PFHM

In which settings is particle filtering better than exact HMM inference?

- Large vs. Small state spaces.
- Prioritizing runtime vs. accuracy.

Particle Filtering works better in large state space settings where we wish to prioritize runtime over accuracy.

Exact HMM inference scales with the state space. For particle filtering, the number of particles determines the time complexity and quality of the approximation. Large (even continuous) state spaces can be approximated by a much smaller number of particles.

Exercise 14.PFNM

Consider an HMM with T timesteps, hidden state variables $X_1, \ldots X_T$, and observed variables $E_1, \ldots E_T$. Let S be the number of possible states for each hidden state variable X. We want to compute (with the forward algorithm) or estimate (with particle filtering) $P(X_T \mid E_1 = e_1, \ldots E_T = e_T)$. How many particles, in terms of S and T, would it take for particle filtering to have the same time complexity as the forward algorithm? You can assume that, in particle filtering, each sampling step can be done in constant time for a single particle (though this is not necessarily the case in reality).

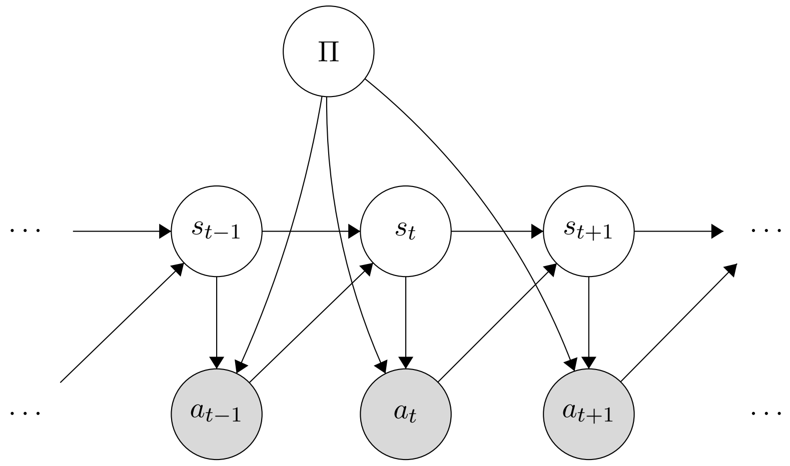


Figure S14.5 Bayes net for Exercise 14.PFDP.

S^2

Exercise 14.PFDP

Consider a modified version of the apprenticeship problem. We are observing an agent's actions in an MDP and are trying to determine which out of a set $\{\pi_1, \ldots, \pi_n\}$ the agent is following. Let the random variable Π take values in that set and represent the policy that the agent is acting under. We consider only *stochastic* policies, so that A_t is a random variable with a distribution conditioned on S_t and Π. As in a typical MDP, S_t is a random variable with a distribution conditioned on S_{t-1} and A_{t-1}. The full Bayes net is shown below.

The agent acting in the environment knows what state it is currently in (as is typical in the MDP setting). Unfortunately, however, we, the observer, cannot see the states S_t. Thus we are forced to use an adapted particle filtering algorithm to solve this problem. Concretely, we will develop an efficient algorithm to estimate $P(\Pi \mid a_{1:t})$.

The Bayes net for this problem is in Figure S14.5.

a. For $t > 10$, what sets of variables need to be given for $S_t \perp\!\!\!\perp S_{t-2}$?

b. We will compute our estimate for $P(\Pi \mid a_{1:t})$ by coming up with a recursive algorithm for computing
$P(\Pi, S_t \mid a_{1:t})$. (We can then sum out S_t to get the desired distribution; in this problem we ignore that step.) Write a recursive expression for $P(\Pi, S_t \mid a_{1:t})$ in terms of the CPTs in the Bayes net above. Hint: Think about the forward algorithm.

c. We now try to adapt particle filtering to approximate this value. Each particle will contain a single state s_t and a potential policy π_i. Write pseudocode for the body of the loop in our adapted particle filtering algorithm. Specifically, write the Elapse Time and Incorporate evidence steps. Remember, we want to approximate $P(\Pi, S_t \mid a_{1:t})$.

a. Using d-separation, we can recover the independence relations as shown.

- $S_t \perp\!\!\!\perp S_{t-2} \mid S_{t-1}, A_{1:t-1}$
- $S_t \perp\!\!\!\perp S_{t-2} \mid \Pi, S_{t-1}$
- $S_t \perp\!\!\!\perp S_{t-2} \mid \Pi, S_{t-1}, A_{1:t-1}$

b. $P(\Pi, S_t \mid a_{1:t}) \propto \sum_{s_{t-1}} P(\Pi, s_{t-1} \mid a_{1:t-1}) P(a_t \mid S_t, \Pi) P(S_t \mid s_{t-1}, a_{t-1})$

c. The pseudocode is as follows.

1. Elapse time: for each particle (s_t, π_i), sample a successor s_{t+1} from $P(S_{t+1} \mid s_t, a_t)$. The policy π' in the new particle is π_i.
2. Incorporate evidence: To each new particle (s_{t+1}, π'), assign weight $P(a_{t+1} \mid s_{t+1}, \pi')$.
3. Resample particles from the weighted particle distribution.

14.5 Kalman Filters

Exercise 14.KFSW

Often, we wish to monitor a continuous-state system whose behavior switches unpredictably among a set of k distinct “modes.” For example, an aircraft trying to evade a missile can execute a series of distinct maneuvers that the missile may attempt to track. A Bayesian network representation of such a **switching Kalman filter** model is shown in Figure **??**.

a. Suppose that the discrete state S_t has k possible values and that the prior continuous state estimate $\mathbf{P}(\mathbf{X}_0)$ is a multivariate Gaussian distribution. Show that the prediction $\mathbf{P}(\mathbf{X}_1)$ is a **mixture of Gaussians**—that is, a weighted sum of Gaussians such that the weights sum to 1.

b. Show that if the current continuous state estimate $\mathbf{P}(\mathbf{X}_t|\mathbf{e}_{1:t})$ is a mixture of m Gaussians, then in the general case the updated state estimate $\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1})$ will be a mixture of km Gaussians.

c. What aspect of the temporal process do the weights in the Gaussian mixture represent?

The results in (a) and (b) show that the representation of the posterior grows without limit even for switching Kalman filters, which are among the simplest hybrid dynamic models.

a. Looking at the fragment of the model containing just S_0, $\mathbf{X}_0$, and $\mathbf{X}_1$, we have

$$\mathbf{P}(\mathbf{X}_1) = \sum_{s_0=1}^{k} P(s_0) \int_{\mathbf{x}_0} P(\mathbf{x}_0)\mathbf{P}(X_1|\mathbf{x}_0, s_0)$$

From the properties of the Kalman filter, we know that the integral gives a Gaussian for each different value of s_0. Hence, the prediction distribution is a mixture of k Gaussians, each weighted by $P(s_0)$.

b. The update equation for the switching Kalman filter is

$$\begin{aligned}
&\mathbf{P}(\mathbf{X}_{t+1}, S_{t+1}|\mathbf{e}_{1:t+1}) \\
&= \alpha \mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1}, S_{t+1}) \sum_{s_t=1}^{k} \int_{\mathbf{x}_t} \mathbf{P}(\mathbf{x}_t, s_t|\mathbf{e}_{1:t})\mathbf{P}(\mathbf{X}_{t+1}, S_{t+1}|\mathbf{x}_t, s_t) \\
&= \alpha \mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1}) \sum_{s_t=1}^{k} P(s_t|\mathbf{e}_{1:t})\mathbf{P}(S_{t+1}|s_t) \int_{\mathbf{x}_t} \mathbf{P}(\mathbf{x}_t|\mathbf{e}_{1:t})\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{x}_t, s_t)
\end{aligned}$$

We are given that $\mathbf{P}(\mathbf{x}_t|\mathbf{e}_{1:t})$ is a mixture of m Gaussians. Each Gaussian is subject to k different linear–Gaussian projections and then updated by a linear-Gaussian observation, so we obtain a sum of km Gaussians. Thus, after t steps we have k^t Gaussians.

c. Each weight represents the probability of one of the k^t sequences of values for the switching variable.

Exercise 14.KALM

Complete the missing step in the derivation of Equation (14.19) on page 499, the first update step for the one-dimensional Kalman filter.

This is a simple exercise in algebra. We have

$$\begin{aligned}
P(x_1|z_1) &= \alpha\, e^{-\frac{1}{2}\left(\frac{(z_1-x_1)^2}{\sigma_z^2}\right)} e^{-\frac{1}{2}\left(\frac{(x_1-\mu_0)^2}{\sigma_0^2+\sigma_x^2}\right)} \\
&= \alpha\, e^{-\frac{1}{2}\left(\frac{(\sigma_0^2+\sigma_x^2)(z_1-x_1)^2+\sigma_z^2(x_1-\mu_0)^2}{\sigma_z^2(\sigma_0^2+\sigma_x^2)}\right)} \\
&= \alpha\, e^{-\frac{1}{2}\left(\frac{(\sigma_0^2+\sigma_x^2)(z_1^2-2z_1x_1+x_1^2)+\sigma_z^2(x_1^2-2\mu_0x_1+\mu_0^2)}{\sigma_z^2(\sigma_0^2+\sigma_x^2)}\right)} \\
&= \alpha\, e^{-\frac{1}{2}\left(\frac{(\sigma_0^2+\sigma_x^2+\sigma_z^2)x_1^2-2((\sigma_0^2+\sigma_x^2)z_1+\sigma_z^2\mu_0)x_1+c}{\sigma_z^2(\sigma_0^2+\sigma_x^2)}\right)} \\
&= \alpha'\, e^{-\frac{1}{2}\left(\frac{(x_1-\frac{(\sigma_0^2+\sigma_x^2)z_1+\sigma_z^2\mu_0}{\sigma_0^2+\sigma_x^2+\sigma_z^2})^2}{(\sigma_0^2+\sigma_x^2)\sigma_z^2/(\sigma_0^2+\sigma_x^2+\sigma_z^2)}\right)} .
\end{aligned}$$

Exercise 14.VARI

Let us examine the behavior of the variance update in Equation (14.20) (page 499).

a. Plot the value of σ_t^2 as a function of t, given various values for σ_x^2 and σ_z^2.

b. Show that the update has a fixed point σ^2 such that $\sigma_t^2 \to \sigma^2$ as $t \to \infty$, and calculate the value of σ^2.

c. Give a qualitative explanation for what happens as $\sigma_x^2 \to 0$ and as $\sigma_z^2 \to 0$.

a. See Figure S14.6.

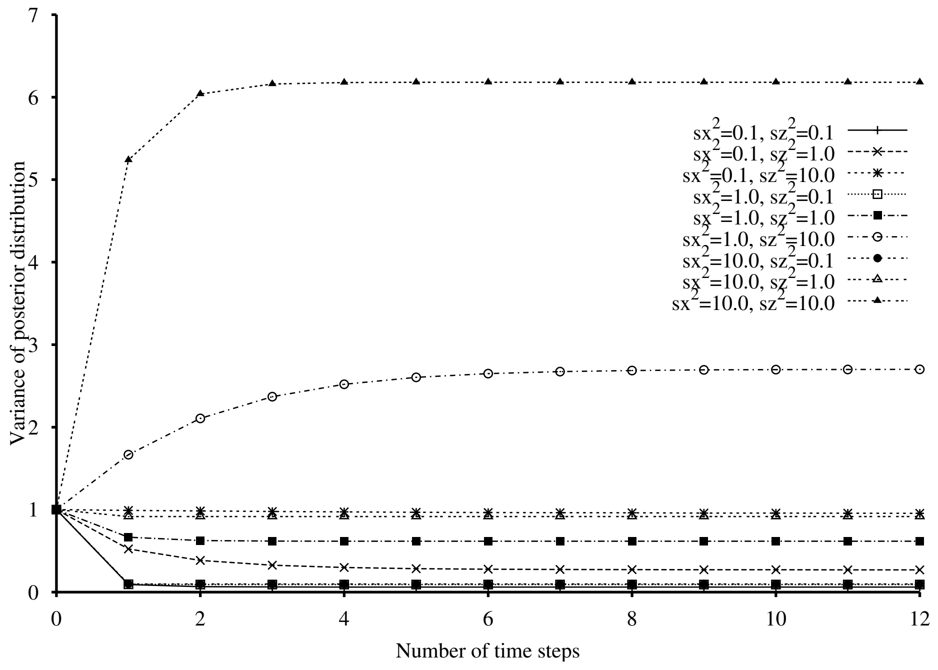


Figure S14.6 Graph for Ex. 15.7, showing the posterior variance σ_t^2 as a function of t for various values of σ_x^2 and σ_z^2.

b. We can find a fixed point by solving

$$\sigma^2 = \frac{(\sigma^2 + \sigma_x^2)\sigma_z^2}{\sigma^2 + \sigma_x^2 + \sigma_z^2}$$

for σ^2. Using the quadratic formula and requiring $\sigma^2 \geq 0$, we obtain

$$\sigma^2 = \frac{-\sigma_x^2 + \sqrt{\sigma_x^4 + 4\sigma_x^2\sigma_z^2}}{\sigma_z^2}$$

We omit the proof of convergence, which, presumably, can be done by showing that the update is a contraction (i.e., after updating, two different starting points for σ_t become closer).

c. As $\sigma_x^2 \to 0$, we see that the fixed point $\sigma^2 \to 0$ also. This is because $\sigma_x^2 = 0$ implies a deterministic path for the object. Each observation supplies more information about this path, until its parameters are known completely.

As $\sigma_z^2 \to 0$, the variance update gives $\sigma^{t+1} \to 0$ immediately. That is, if we have an exact observation of the object's state, then the posterior is a delta function about that observed value regardless of the transition variance.

14.6 Dynamic Bayesian Networks

Exercise 14.VACP

We have described three policies for the vacuum robot: (1) a uniform random walk, (2) a bias for wandering southeast, as described in Exercise 14.HMMR, and (3) the policy described in Exercise 14.ROOM. Suppose an observer is given the observation sequence from a vacuum robot, but is not sure which of the three policies the robot is following. What approach should the observer use to find the most likely path, given the observations? Implement the approach and test it. How much does the localization accuracy suffer, compared to the case in which the observer knows which policy the robot is following?

The correct model is a DBN with location and heading variables along with a single atemporal variable for the policy Φ which has all the state variables as children. We can reduce the inference problem to computations on the three original models by summing out the policy variable as follows:

$$\begin{aligned} P(X_t \mid e_{1:t}) &= \sum_{\phi} P(X_t \mid e_{1:t}, \phi) P(\phi \mid e_{1:t}) \\ &= \alpha \sum_{\phi} P(X_t \mid e_{1:t}, \phi) P(e_{1:t} \mid \phi) P(\phi) \end{aligned}$$

Thus, we need to run both the filtering calculation and the likelihood calculation (as described in the text) for each of the three policies. The likelihood times the prior gives the posterior for the policy, and the state estimate is the posterior-weighted combination of the three state estimates.

Generally we expect the policy to be quickly recognizable in cases where it has a significant impact on the percepts; if the impact is small (as, for example, in the case of a small southeasterly bias) the inability to identify the policy will not have a large effect on accuracy.

Exercise 14.SLEP

A professor wants to know if students are getting enough sleep. Each day, the professor observes whether the students sleep in class, and whether they have red eyes. The professor has the following domain theory:

- The prior probability of getting enough sleep, with no observations, is 0.7.
- The probability of getting enough sleep on night t is 0.8 given that the student got enough sleep the previous night, and 0.3 if not.
- The probability of having red eyes is 0.2 if the student got enough sleep, and 0.7 if not.
- The probability of sleeping in class is 0.1 if the student got enough sleep, and 0.3 if not.

Formulate this information as a dynamic Bayesian network that the professor could use to filter or predict from a sequence of observations. Then reformulate it as a hidden Markov model that has only a single observation variable. Give the complete probability tables for the model.

The DBN has three variables: S_t, whether the student gets enough sleep; R_t, whether they have red eyes in class; C_t, whether the student sleeps in class. S_t is a parent of S_{t+1}, R_t, and C_t. The CPTs are given by

$$\begin{aligned}
P(s_0) &= 0.7\\
P(s_{t+1}|s_t) &= 0.8\\
P(s_{t+1}|\neg s_t) &= 0.3\\
P(r_t|s_t) &= 0.2\\
P(r_t|\neg s_t) &= 0.7\\
P(c_t|s_t) &= 0.1\\
P(c_t|\neg s_t) &= 0.3
\end{aligned}$$

To reformulate as an HMM with a single observation node, simply combine the 2-valued variables "having red eyes" and "sleeping in class" into a single 4-valued variable, multiplying together the emission probabilities. (Probability tables omitted.)

Exercise 14.SLER

For the DBN specified in Exercise 14.SLE and for the evidence values

$\mathbf{e}_1 =$ not red eyes, not sleeping in class
$\mathbf{e}_2 =$ red eyes, not sleeping in class
$\mathbf{e}_3 =$ red eyes, sleeping in class

perform the following computations:

a. State estimation: Compute $P(EnoughSleep_t|\mathbf{e}_{1:t})$ for each of $t = 1, 2, 3$.
b. Smoothing: Compute $P(EnoughSleep_t|\mathbf{e}_{1:3})$ for each of $t = 1, 2, 3$.
c. Compare the filtered and smoothed probabilities for $t = 1$ and $t = 2$.

a. We apply the forward algorithm to compute these probabilities.

$$\begin{aligned}
P(S_0) &= \langle 0.7, 0.3\rangle \\
P(S_1) &= \sum_{s_0} P(S_1|s_0)P(s_0) \\
&= (\langle 0.8, 0.2\rangle 0.7 + \langle 0.3, 0.7\rangle 0.3) \\
&= \langle 0.65, 0.35\rangle \\
P(S_1|e_1) &= \alpha\, P(e_1|S_1)P(S_1) \\
&= \alpha\,\langle 0.8\times 0.9, 0.3\times 0.7\rangle\langle 0.65, 0.35\rangle \\
&= \alpha\,\langle 0.72, 0.21\rangle\langle 0.65, 0.35\rangle \\
&= \langle 0.8643, 0.1357\rangle \\
P(S_2|e_1) &= \sum_{s_1} P(S_2|s_1)P(s_1|e_1) \\
&= \langle 0.7321, 0.2679\rangle \\
P(S_2|e_{1:2}) &= \alpha\, P(e_2|S_2)P(S_2|e_1) \\
&= \langle 0.5010, 0.4990\rangle \\
P(S_3|e_{1:2}) &= \sum_{s_2} P(S_3|s_2)P(s_2|e_{1:2}) \\
&= \langle 0.5505, 0.4495\rangle \\
P(S_3|e_{1:3}) &= \alpha\, P(e_3|S_3)P(S_3|e_{1:2}) \\
&= \langle 0.1045, 0.8955\rangle
\end{aligned}$$

Similar to many students during the course of the school term, the student observed here seems to have a higher likelihood of being sleep deprived as time goes on!

b. First we compute the backwards messages:

$$\begin{aligned}
P(e_3|S_3) &= \langle 0.2\times 0.1, 0.7\times 0.3\rangle \\
&= \langle 0.02, 0.21\rangle \\
P(e_3|S_2) &= \sum_{s_3} P(e_3|s_3)P(|s_3)P(s_3|S_2) \\
&= \langle 0.02\times 0.8 + 0.21\times 0.2, 0.02\times 0.3 + 0.21\times 0.7\rangle \\
&= \langle 0.0588, 0.153\rangle \\
P(e_{2:3}|S_1) &= \sum_{s_2} P(e_2|s_2)P(e_3|s_2)P(s_2|S_1) \\
&= \langle 0.0233, 0.0556\rangle
\end{aligned}$$

Then we combine these with the forwards messages computed previously and normalize:

$$\begin{aligned}
P(S_1|e_{1:3}) &= \alpha\, P(S_1|e_1)P(e_{2:3}|S_1) \\
&= \langle 0.7277, 0.2723\rangle \\
P(S_2|e_{1:3}) &= \alpha\, P(S_2|e_{1:2})P(e_3|S_1) \\
&= \langle 0.2757, 0.7243\rangle \\
P(S_3|e_{1:3}) &= \langle 0.1045, 0.8955\rangle
\end{aligned}$$

c. The smoothed analysis places the time the student started sleeping poorly one step earlier than than filtered analysis, integrating future observations indicating lack of sleep at the last step.

Exercise 14.SLES

Suppose that a particular student shows up with red eyes and sleeps in class every day. Given the model described in Exercise 14.SLEP, explain why the probability that the student had enough sleep the previous night converges to a fixed point rather than continuing to go down as we gather more days of evidence. What is the fixed point? Answer this both numerically (by computation) and analytically.

The probability reaches a fixed point because there is always some chance of spontaneously starting to sleep well again, and students who sleep well sometimes have red eyes and sleep in class. Even if we knew for sure that the student didn't sleep well on day t, and that they slept in class with red eyes on day $t+1$, there would still be a chance that they slept well on day $t+1$.

Numerically one can repeatedly apply the forward equations to find equilibrium probabilities of $\langle 0.0432, 0.9568\rangle$.

Analytically, we are trying to find the vector $(p_0, p_1)^T$ which is the fixed point to the forward equation, which one can pose in matrix form as

$$(p_0, p_1)^T = \alpha \begin{pmatrix} 0.016 & 0.006 \\ 0.042 & 0.147 \end{pmatrix} (p_0, p_1)^T$$

where α is a normalization constant. That is, $(p_0, p_1)^T$ is an eigenvector of the given matrix. Computing, we find that the only positive eigenvalue is 0.1487, which has eigenvector (normalized to sum to one) $(0.0432, 0.9568)^T$, just as we numerically computed.

Exercise 14.BATS

This exercise analyzes in more detail the persistent-failure model for the battery sensor in Figure 14.15(a) (page 507).

a. Figure 14.15(b) stops at $t = 32$. Describe qualitatively what should happen as $t \to \infty$ if the sensor continues to read 0.

b. Suppose that the external temperature affects the battery sensor in such a way that transient failures become more likely as temperature increases. Show how to augment the

DBN structure in Figure 14.15(a), and explain any required changes to the CPTs.

c. Given the new network structure, can battery readings be used by the robot to infer the current temperature?

a. The curve of interest is the one for $E(Battery_t|\ldots 5555000000\ldots)$. In the absence of any useful sensor information from the battery meter, the posterior distribution for the battery level is the same as the projection without evidence. The transition model for the battery includes a small probability for downward transitions in the battery level at each time step, but zero probability for upward transitions (there are no recharging actions in the model). Thus, the stationary distribution towards which the battery level tends has value 0 with probability 1. The curve for $E(Battery_t|\ldots 5555000000\ldots)$ will asymptote to 0.

b. See Figure S14.7. The CPT for $BMeter_1$ has a probability of transient failure (i.e., reporting 0) that increases with temperature.

c. The agent can obviously calculate the posterior distribution over $Temp_t$ by filtering the observation sequence in the usual way. This posterior can be informative if the effect of temperature on transient failure is non-negligible and transient failures occur more frequently than do major changes in temperature. Essentially, the temperature is estimated from the frequency of “blips” in the sequence of battery meter readings.

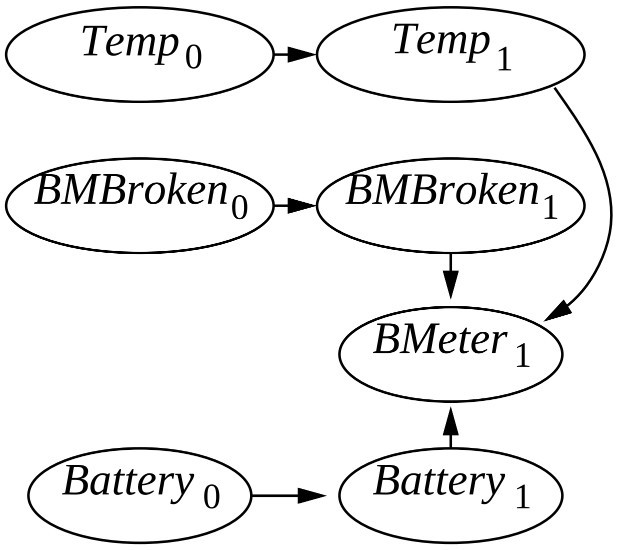


Figure S14.7 Modification of Figure 15.13(a) to include the effect of external temperature on the battery meter.

Exercise 14.BATT

Consider the DBN in Figure 14.13(b). In the chapter, the battery level $Battery_t$ and the battery meter reading $BMeter_t$ are assumed to be integer-valued with a range of 0 to 5. In this exercise, we will look at a more realistic model where they are continuous variables; for simplicity we will assume a range $[0,1]$.

a. Give exact expressions for the sensor model $P(BMeter_t \mid Battery_t)$, for both beta distributions and truncated normal distributions, where the mode is at the true value and the standard deviation is 0.1. (Details of these distributions are available in many online sources.) Plot the conditional distributions for $Battery_t = 0.0,\ 0.2,\ 0.4,\ 0.6,\ 0.8,\ 1.0$.

b. Describe in detail a suitable distribution for the transition model $P(Battery_{t+1} \mid Battery_t, \dot{\mathbf{X}}_t)$, where $\dot{\mathbf{X}}_t$ is a two-dimensional velocity vector. You may assume that the battery drains at a small constant rate r plus an amount proportional to the absolute velocity of the robot, with a standard deviation that is also proportional to the absolute velocity. Remember that the battery charge cannot go below zero and cannot increase.

c. Explain how to integrate a $Charging_t$ variable into the DBN, which is true just when the robot is plugged into the charging station.

d. What might a reasonable prior $P(Battery_0)$ look like?

[[TBC]]

Exercise 14.DBNE

Consider applying the variable elimination algorithm to the umbrella DBN unrolled for three slices, where the query is $\mathbf{P}(R_3|u_1,u_2,u_3)$. Show that the space complexity of the algorithm—the size of the largest factor—is the same, regardless of whether the rain variables are eliminated in forward or backward order.

The process works exactly as on page 525. We start with the full expression:

$$\mathbf{P}(R_3|u_1,u_2,u_3) = \alpha \sum_{r_1} \sum_{r_2} P(r_1)P(u_1|r_1)P(r_2|r_1)P(u_2|r_2)\mathbf{P}(R_3|r_2)\mathbf{P}(u_3|R_3)$$

Whichever order we push in the summations, the variable elimination process never creates factors containing more than two variables, which is the same size as the CPTs in the original network. In fact, given an HMM sequence of arbitrary length, we can eliminate the state variables in any order.

Exercise 14.RBPF

Consider the Bayes net obtained by unrolling the DBN in Figure 14.20 to time step t.

Use the conditional independence properties of this network to show that

$$\begin{aligned}&\mathbf{P}(Dirt_{1,0:t},\ldots,Dirt_{42,0:t}\mid DirtSensor_{1:t}, WallSensor_{1:t}, Location_{1:t})\\ &\quad = \prod_i \mathbf{P}(Dirt_{i,0:t}\mid DirtSensor_{1:t}, Location_{1:t})\,.\end{aligned}$$

$$\begin{aligned}&\mathbf{P}(Dirt_{1,0:t},\ldots,Dirt_{42,0:t}\mid DirtSensor_{1:t}, WallSensor_{1:t}, Location_{1:t})\\ &\quad = \mathbf{P}(Dirt_{1,0:t},\ldots,Dirt_{42,0:t}\mid DirtSensor_{1:t}, Location_{1:t})\\ &\quad = \mathbf{P}(Dirt_{1,0:t}\mid DirtSensor_{1:t}, Location_{1:t})\\ &\qquad \mathbf{P}(\ldots\mid DirtSensor_{1:t}, Location_{1:t})\\ &\qquad \mathbf{P}(Dirt_{42,0:t}\mid DirtSensor_{1:t}, Location_{1:t})\\ &\quad = \prod_i \mathbf{P}(Dirt_{i,0:t}\mid DirtSensor_{1:t}, Location_{1:t})\,.\end{aligned}$$

Exercise 14.RAOB

Consider a probability model $\mathbf{P}(\mathbf{X}, \mathbf{Y}, Z, \mathbf{E})$, where Z is a single query variable and evidence $\mathbf{E}=\mathbf{e}$ is given. A basic Monte Carlo algorithm generates N samples (ideally) from $\mathbf{P}(\mathbf{X}, \mathbf{Y}, Z\mid \mathbf{E}=\mathbf{e})$ and estimates the query probability $P(Z=z\mid \mathbf{E}=\mathbf{e})$ from those samples. This gives an unbiased estimate but the variance may be quite large. The basic idea of **Rao-Blackwellization** in this context is to generate N samples of, say, $(\mathbf{X}, Z)$ and, for each sample $\mathbf{x}_j, z_j$, to perform exact inference for $\mathbf{P}(\mathbf{Y}\mid \mathbf{x}_j, z_j, \mathbf{e})$. Explain how this yields an estimate for the query $P(Z=z\mid \mathbf{E}=\mathbf{e})$ and show that the variance of the estimate is no larger than that from the original non-Rao-Blackwellized procedure.

TBC

Exercise 14.DRGT

In California, whether it rains or not from each day to the next forms a Markov chain (note: this is a terrible model for real weather). However, sometimes California is in a drought and sometimes it is not. Whether California is in a drought from each day to the next itself forms a Markov chain, and the state of this Markov chain affects the transition probabilities in the rain-or-shine Markov chain. The state diagram for droughts, rain given drought, and rain given no drought appear in Figure S14.8.

a. Draw a dynamic Bayes net which encodes this behavior. Use variables D_{t-1}, D_t, D_{t+1}, R_{t-1}, R_t, and R_{t+1}. Assume that on a given day, it is determined whether or not there is a drought before it is determined whether or not it rains that day.

b. Draw the CPT for D_t in the above DBN. Fill in the actual numerical probabilities.

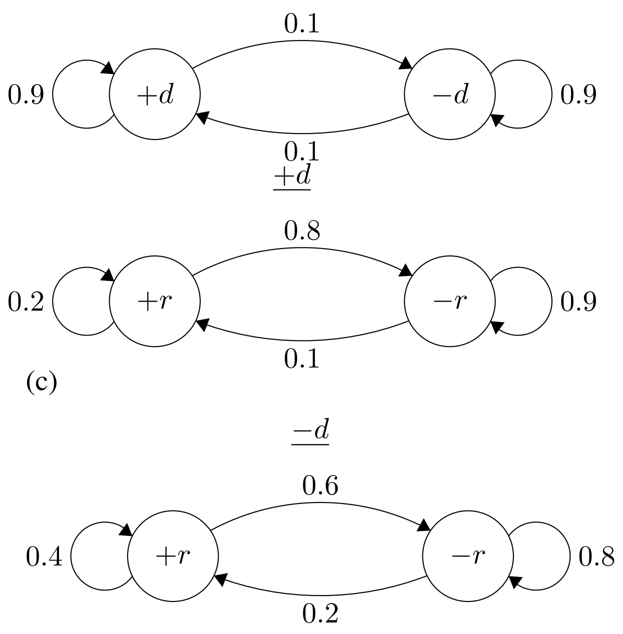


Figure S14.8 State diagrams for drought, (+d) rain given drought, and (-d) rain given no drought.

---

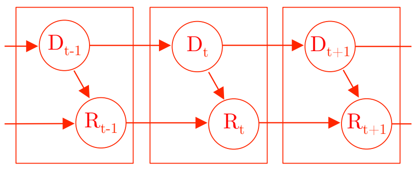
c. Draw the CPT for R_t in the above DBN. Fill in the actual numerical probabilities.

Suppose we are observing the weather on a day-to-day basis, but we cannot directly observe whether California is in a drought or not. We want to predict whether or not it will rain on day $t+1$ given observations of whether or not it rained on days 1 through t.

d. First, we need to determine whether California will be in a drought on day $t+1$. Derive a formula for $P(D_{t+1}|r_{1:t})$ in terms of the given probabilities (the transition probabilities on the above state diagrams) and $P(D_t|r_{1:t})$ (that is, you can assume we've already computed the probability there is a drought today given the weather over time).

e. Now derive a formula for $P(R_{t+1}|r_{1:t})$ in terms of $P(D_{t+1}|r_{1:t})$ and the given probabilities.

a. The DBN can be constructed as follows.



b. Reading off the probabilities of the markov chain into a CPT, we get:

$P(D_t \mid D_{t-1})$		
$+d_{t-1}$	$+d_t$	0.9
$+d_{t-1}$	$-d_t$	0.1
$-d_{t-1}$	$+d_t$	0.1
$-d_{t-1}$	$-d_t$	0.9

c. Reading off the probabilities of the markov chain into a CPT, we get:

$P(R_t \mid R_{t-1}, D_t)$			
$+d_t$	$+r_{t-1}$	$+r_t$	0.2
$+d_t$	$+r_{t-1}$	$-r_t$	0.8
$+d_t$	$-r_{t-1}$	$+r_t$	0.1
$+d_t$	$-r_{t-1}$	$-r_t$	0.9
$-d_t$	$+r_{t-1}$	$+r_t$	0.4
$-d_t$	$+r_{t-1}$	$-r_t$	0.6
$-d_t$	$-r_{t-1}$	$+r_t$	0.2
$-d_t$	$-r_{t-1}$	$-r_t$	0.8

d. $P(D_{t+1} \mid r_{1:t}) = \sum_{d_t} P(D_{t+1} \mid d_t) P(d_t \mid r_{1:t})$

e. $P(R_{t+1} \mid r_{1:t}) = \sum_{d_{t+1}} P(d_{t+1} \mid r_{1:t}) P(R_{t+1} \mid r_t, d_{t+1})$