Contents

1 Bayes Nets

Motivation

We generalize Markov chains to Bayes nets, which gives us a way to simulate robots complete with sensor observations. We will view simulation as drawing samples from a structured probability distribution, described by bayes nets. The probability distribution can be viewed as a stochastic world model, by describing how actions affect the state, and how how sensors reflect state.

The data structures we build below can be used to control and plan for a real robot. The probabilistic nature of sensor measurements will manifest itself in being able to make probabilistic statements about the robot's state, in a process called inference. In this section we will do some of that, but will mostly be content with the modeling part.

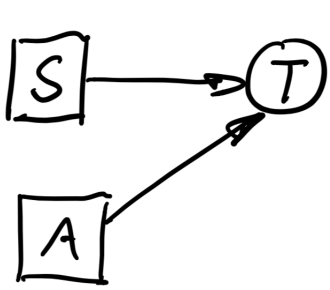
1.1 A Grid-world Agent

Our running example will be a robot that exists on a plane, and we will describe the robot's location with the 2D coordinate of a 10 by 10 grid that discretizes the space. To describe the location or state S of the robot we use a 10×10 grid, an as such there are 100 possible outcomes. This is a much larger state space than the robot vacuum cleaner example we used before, but still manageable. The robot has a sensor to figure out where it is, and it's action space is defined by moving left, right, up, or down in the grid.

				.1					
	.1								
			.1						
.1					.1			.1	
		.1				.1			
							.1		
									.1

Table 1: A probability mass function (PMF), describing where a robot might be in grid-world.

As we discussed, a probability mass function or PMF for the state S of the robot assigns a probability $p_{i,j}$ to each grid cell (i, j), and can encode our prior knowledge about where the robot could be, given no other information. We can use whatever symbol we want for the state, so here we will use S. We deliberately use row index i and column index j, so we can think of a probability mass function (PMF) as a matrix. Some examples for such a PMF include: uniform, a single particular starting position, more probable to be on the left, etc... An example is shown in Table 1, which models that given no other information, we expect to find the robot in one of 10 different possible locations with equal probability.



(a) State transition model $P(T|S, A)$

(b) Sensor model $P(O|S)$

Figure 1: Conditional distributions to model acting and sensing.

As we saw in the previous section, we can model the outcome of **actions** using conditional probability distributions. In particular, we need a conditional probability distribution

$$P(T|S = s, A = a)$$

on the next state T, given the value s of the current state S, and the value a of the action A. The corresponding graphical model fragment is shown in Figure 1a.

As we discussed, conditional probabilities do not have to be specified as giant tables. In case we index the discrete states with semantically meaningful indices, as in the grid-world example, we can often use a parametric form. In fact, because the state space is potentially quite large such a **state transition model** is almost never explicitly specified, but rather exploits the semantics of the states and actions to provide a more compact representation of the conditional distribution.

Exercises

1. Specify a parametric conditional distribution for the action models in the grid-world that is somewhat realistic, yet not completely deterministic.

2. It is possible to create models that do not reflect everyday physics. For example, how could we model the game "Portal"?

1.2 Modeling Sensors

Conditional probability distributions are also a great way to model **sensors**. We can use a random variable O to model an observation, and specify via a distribution $P(O|S = s)$ how the sensor behaves, given that we are in a particular state s. This is illustrated in Figure 1b. A complete **sensor model** specifies this in a (giant) CPT for every possible state. An observation O can be

rather impoverished, or very detailed, and one can also envision modeling several different sensors on the robot. In the latter case, we will be able to fuse the information from multiple sensors.

Again, we can often give specify this sensor model in parametric form. For example, a simple sensor would simply report the horizontal coordinate j of the robot faithfully, in which case we have

$$\begin{cases} P(O = k|S = i, j) = 1 & \text{iff } k = j \\ P(O = k|S = i, j) = 0 & \text{otherwise} \end{cases}$$

However, the power of using probability distributions comes from the fact that we can model less accurate/faithful sensing. For example, here is a parametric model for a sensor that reports the vertical coordinate i of the robot, but with 9% probability gives a random faulty reading:

$$\begin{cases} P(O = k|S = i, j) = 0.91 & \text{iff } k = i \\ P(O = k|S = i, j) = 0.01 & \text{otherwise} \end{cases} \tag{1}$$

Exercise

How would you code a parametric conditional distribution for a sensor that only approximately reflects location. Hint: you could use Manhattan distance to define a notion of "error".

1.3 Joint Distribution

Given that we can model actions and sensors as conditional probability distributions and visualize them with small graphical snippets, we can ask what happens when we start combining these local models. An example was given in the previous chapter, where we discussed Markov chains. Below we will formalize the notion more, starting with the concept of joint probability and then generalizing to Bayes nets in the next section.

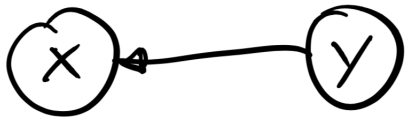


Figure 2: Joint probability distribution on two variables X and Y.

The notion of a joint distribution formalizes what happens when the condition in a conditional probability is itself a random variable. In other words, what happens if we consider the parameter Y in a conditional probability distribution $P(X|Y)$ not as a known parameter but as a random variable itself, with probability distribution $P(Y)$? In this case, we can define the **joint probability distribution** over the pair of variables X and Y, given by the **chain rule**:

$$P(X, Y) = P(X|Y)P(Y)$$

Graphically, we have the graph fragment shown in Figure 2.

To sample from a joint distribution $P(X, Y)$ we can take the graph as a guide, just like we did Markov chains: we first sample a value y from $P(Y)$, as it does not depend on any other variable, and then sample a value x from the conditional distribution $P(X|Y = y)$.

Given a joint distribution $P(X, Y)$ we can ask what the probability is of an outcome x for X, irrespective of the value of Y. We call this the **marginal probability distribution** of X, and it can be calculated as

$$P(X) = \sum_y P(X, Y = y)$$

and of course we can also calculate the marginal distribution of Y, in the same way:

$$P(Y) = \sum_x P(X = x, Y)$$

We can also calculate the conditional distributions when given the joint distribution, by taking the joint probability distribution and dividing by the appropriate marginal:

$$P(X|Y) = P(X, Y)/P(Y)$$
$$P(Y|X) = P(X, Y)/P(X)$$

This is really just the chain rule, re-arranged.

Exercises

1. Viewing the integer coordinates i and j as separate random variables, use the distribution $P(i, j) = P(S)$ from Table 1 and calculate the marginals $P(i)$ and $P(j)$.
2. Similarly, what do the conditional probability tables for $P(i|j)$ and $P(j|i)$ look like? Before you start calculating, ponder the relationship with Table 1.

1.4 Bayes Nets

Bayes nets generalize the notion of a joint probability distribution defined by a graph to an arbitrary number of variables, connected with directed edges. Above we used directed edges to indicate conditional links between random variables, and here we do exactly the same. However, to make this useful we need a careful definition, given below.

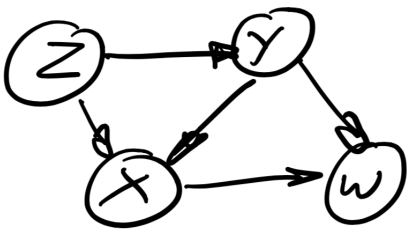


Figure 3: A Bayes net representing a more general joint distribution over the random variables W,X,Y, and Z.

A **Bayes net** is a directed acyclic graph (DAG) describing a factored probability distribution a set of random variables. The joint distribution on the set of all variables is given as

$$P(\{X_i\}) = \prod_{i=1}^{n} P(X_i|\Pi_i)$$

where n is the number of variables, and Π_i denotes the set of parents for variable X_i. An example of a Bayes net is shown in Figure 3, and it is simply a graphical representation of which random variables's CPT depend on which other variables. In this case, the joint distribution is

$$P(W, X, Y, Z) = P(W|X, Y)P(X|Y, Z)P(Y|Z)P(Z).$$

Note that the order in which we multiply the conditionals does not matter.

CPT	# entries
$P(Z)$	9
$P(Y\|Z)$	90
$P(X\|Y, Z)$	900
$P(W\|X, Y)$	900

Table 2: Number of entries in each CPT for the Bayes net of Figure 3 (10 outcomes per variable).

A Bayes net can be a very efficient representation of complex probability distributions, as they encode the dependence and especially independence relationships between the variables. For example, if we were to construct a full table of probabilities for each possible outcome of the variables W,X,Y, and Z, the table could be quite long. For example, if we assume they all have 10 possible values, then the full joint has 10^4 entries, i.e., $10,000$ unique values. You can save a tiny bit, because they have to sum up to 1, so strictly speaking we need only $9,999$ values. In contrast, we can tally how many entries all four CPT tables have for the Bayes net in Figure 3. In Table 2 we did just that; for example, $P(X|Y, Z)$ has 900 entries, i.e., 9 (independent) entries for each of 100 possible combinations of Y and Z. Hence, the total number of parameters we need is only $1,899$, which is rather less than $9,999$.

1.5 Ancestral Sampling

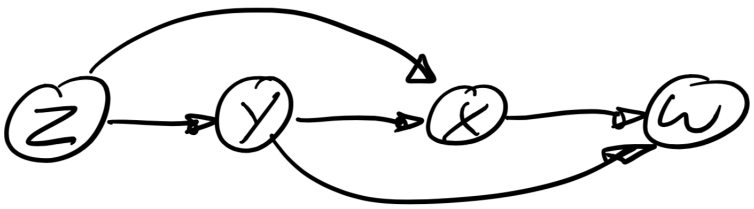


Figure 4: The topological sort of the Bayes net in Figure 3.

Sampling from the joint distribution given in Bayes net form can be done by sampling each variable in turn, but making sure that we always sample a node's parents first. This can be done through the notion of a **topological sort** of the DAG.

An easy algorithm to obtain a topological sort is Kahn's algorithm, which recursively removes a node from the graph that either has no parents, or whose parents have all been removed already. The order in which nodes were removed constitutes a (non-unique) topological sort order.

Sampling is then done by inverse transform sampling for each variable separately, in topological sort order. An example is shown in Figure 4, which shows a topological sort of the Bayes net in Figure 3. Hence, in this example we sample first Z, then Y, then X, and then W. Note that in this case the topological sort is unique, but that is an exception rather than the rule.

1.6 Dynamic Bayes Nets and Simulation

Note that directed cycles are not allowed in a Bayes net, i.e., the graph is acyclic. Hence, one might wonder how we deal with time: if a robot is all about the sense-think-act cycle, would we not expect a cycle in the graph when describing robots? The answer is to unroll time, as we discuss below.

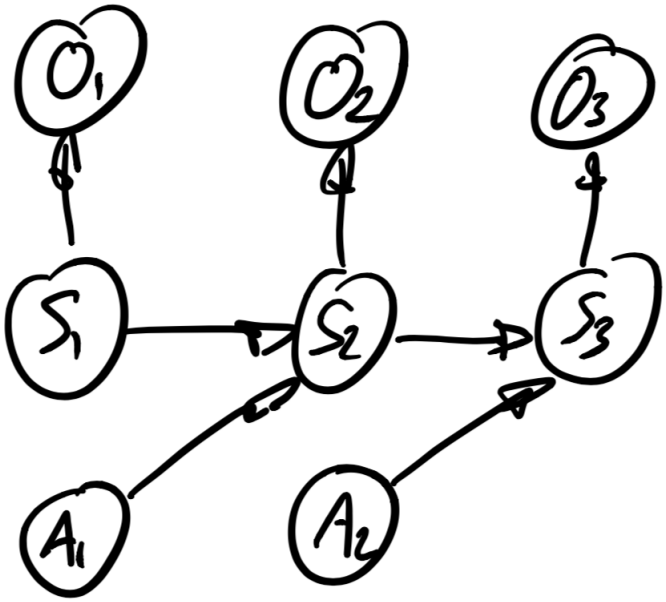


Figure 5: A Bayes net modeling our robot example.

When a Bayes net is used to represent the evolution of a system or agent over time, we call it a **dynamic Bayes net** or DBN. A small DBN fragment for a generic robot, modeled using discrete states, observations, and actions, is shown in Figure 5. We recognize subgraphs modeling the sensor behavior, and the effects of actions, at each time step. Hence, this generalizes Markov chains.

Simulation of the robot is then equivalent to sampling from this DBN, and in this case the topological sort is rather obvious, and hence so is the simulation algorithm:

1. First, sample the initial state s_1 from $P(S_1)$, a prior over the state. Set $k = 1$.
2. Next, simulate the sensor by sampling from the sensor model $P(O_k|S_k = s_k)$, yielding o_k.
3. Then, sample an action a_k from $P(A_k)$.
4. Lastly, simulate the effect of this action by sampling the next state s_k from

$$P(S_{k+1}|S_k = s_k, A_k = a_k).$$

5. Increase k and return to step 2.

Exercise

Simulate two different realizations from the dynamic Bayes net in Figure 5.

1.7 Bayes' Rule

We now derive Bayes' rule, which allows us to infer knowledge about a variable, say the robot state S, given an observed sensor value o. The rule is named after the reverend Thomas Bayes, an eighteenth century minister who took an interest in probability later in life. He also lends his name to the Bayes nets which we discussed in Section 1.4.

Bayes' rule allows us to calculate the **posterior probability distribution** $P(S|O = o)$ on the variable S given an observed value for O. In Bayes' rule we use the term "prior" to indicate knowledge we have about a variable S before seeing evidence for it, and use "posterior" to denote the knowledge after having incorporated evidence. In our case, the evidence is the observed value o. To calculate the posterior, the elements we need are a prior probability distribution $P(S)$, a conditional probability distribution $P(O|S)$ modeling the measurement, and the value o itself. Given these elements, we can calculate the posterior probability distribution on S:

$$P(S|O = o) = \frac{P(O = o|S)P(S)}{P(O = o)} \tag{2}$$

The proof is simple and involves applying the chain rule in two ways: $P(O|S)P(S) = P(S, O) = P(S|O)P(O)$.

Because in the above the observation is known, statisticians introduce a different quantity called the **likelihood function**,

$$L(S; o) \stackrel{\Delta}{=} P(O = o|S)$$

which is a function of S, and reads as "the likelihood of the state S given the observed measurement o". Strictly speaking, any function proportional to $P(O = o|S)$ will do as the likelihood, but the above definition is simpler.

In addition, note that in Equation 2 the quantity $P(O = o)$ simply acts as a normalization constant. Given these two facts, we can state Bayes' rule in a more intuitive way - and easier to remember - as

$$P(S|O = o) \propto L(S; o)P(S) \tag{3}$$

or "the posterior is proportional to the likelihood weighted by the prior."

Finally, when actually computing a posterior there is not even a need to think about anything but the joint distribution. Indeed, because by the chain rule we have $P(O = o|S)P(S) = P(S, O = o)$, and because $P(O = o)$ is just a normalization factor, we obtain a third form of Bayes' rule, which is the simplest of all:

$$P(S|O = o) \propto P(S, O = o) \tag{4}$$

Hence, if we are given a formula or table of joint probability entries $P(S, O)$, it suffices to just select all of them where $O = o$, normalize, and voila!

Exercises

1. Apply Bayes' rule to calculate the posterior probability $P(S|O = 5)$, using the prior $P(S)$ from Table 1 and the sensor model specified in Equation 1.

2. Suppose we are interested in estimating the probability of an obstacle in front of the robot, given an "obstacle sensor" that is accurate 90% of the time. Apply Bayes' rule to this example by creating the sensor model, prior, and deriving the posterior.

3. For the previous example, what happens if the sensor is random, i.e., it just outputs "obstacle detected" with 50% probability?

1.8 Inference in Bayes Nets

Inference is the process of obtaining knowledge about a subset of variables given the known values for another subset of variables. In this section we will talk about how to do inference when the joint distribution is specified using a Bayes net, but we will not take advantage of the sparse structure of the network. Hence, the algorithms below are completely general, for any (discrete) joint probability distribution, as long as you can evaluate the joint.

The simplest case occurs when we can partition the variables into two sets: the hidden variables $\mathcal{X}$ and the observed values $\mathcal{Z}$. Then we can simply apply Bayes' rule, but now applied to sets of variables, to obtain an expression for the posterior over the hidden variables $\mathcal{X}$. Using the easy version of Bayes' rule from equation 4 we obtain

$$P(\mathcal{X}|\mathcal{Z} = \mathfrak{z}) \propto P(\mathcal{X}, \mathcal{Z} = \mathfrak{z}),$$

where $\mathfrak{z}$ is the set of observed values for all variables in $\mathcal{Z}$.

x	y	$P(W = 2, X = x, Y = y, Z = 7)$
1	1	$P(W = 2\|X = 1, Y = 1)P(X = 1\|Y = 1, Z = 7)P(Y = 1\|Z = 7)P(Z = 7)$
1	2	$P(W = 2\|X = 1, Y = 2)P(X = 1\|Y = 2, Z = 7)P(Y = 2\|Z = 7)P(Z = 7)$
$\vdots$	$\vdots$	$\vdots$
10	9	$P(W = 2\|X = 10, Y = 9)P(X = 10\|Y = 9, Z = 7)P(Y = 9\|Z = 7)P(Z = 7)$
10	10	$P(W = 2\|X = 10, Y = 10)P(X = 10\|Y = 10, Z = 7)P(Y = 10\|Z = 7)P(Z = 7)$
		$\sum_{x,y} P(W = 2, X = x, Y = y, Z = 7)$

Table 3: Computation of the posterior $P(X, Y|W = 2, Z = 7)$ by enumeration. Note that all entries have to be normalized by the sum at the bottom to get the correct, normalized posterior.

There is an easy algorithm to calculate the posterior distribution above: simply enumerate all tuples $\mathcal{X}$ in a table, evaluate $P(\mathcal{X}, \mathcal{Z} = \mathfrak{z})$ for each one, and then normalize. As an example, let us consider the Bayes net from Figure 3, and take $\mathcal{X} = (X, Y)$ and $\mathcal{Z} = (W, Z)$. As before, let us assume that each variable can take on 10 different outcomes, and that $\mathfrak{z} = (2, 7)$. The resulting table for $P(X, Y|W = 2, Z = 7) \propto P(W = 2, X, Y, Z = 7)$ is shown in Table 3.

The simple algorithm outlined above is not efficient. In the example the table is 100 entries long, and in general the number of entries is exponential in the size of $\mathcal{X}$. However, when inspecting the entries in Table 3 there are already some obvious ways to save: for example, $P(Z = 7)$ is a common factor in all entries, so clearly we should not even bother multiplying it in. In the next section, we will discuss methods to fully exploit the structure of the Bayes net to perform efficient inference.

Exercises

1. Show that in the example from Figure 3, if we condition on known values for $\mathcal{Z} = (X, Y)$, the posterior $P(W, Z|X, Y)$ factors, and as a consequence we only have to enumerate two tables of length 10, instead of a large table of size 100.

2. Calculate the size of the table needed to enumerate the posterior over the states S the Bayes net from Figure 5, given the value of all observations O and actions A.

3. Show that if we are given the states, inferring the actions is actually quite efficient, even with the brute force enumeration. Hint: this is similar to the first exercise above.

1.9 Finding the Most Probable Explanation

A common inference problem associated with Bayes nets is the **most probable explanation** or MPE for $\mathcal{X}$: given the values $\mathfrak{z}$ for $\mathcal{Z}$, what is the most probable joint assignment to the other variables $\mathcal{X}$? While the posterior gives us the complete picture, the MPE is different in nature: it is a single assignment of values to $\mathcal{X}$. For example, given $\mathfrak{z} = (2, 7)$, the MPE for $\mathcal{X}$ could be $X = 3$ and $Y = 6$. Note that to compute the MPE, we need not bother with normalizing: we can simply find the maximum entry in the unnormalized posterior values.

If we had an efficient way to do inference, an MPE estimate would be a great way to estimate the trajectory of a robot over time. For example, using the dynamic Bayes net example from Figure 5, assume that we are given the value of all observations O and actions A. Then the MPE would simply be a trajectory of robot states through the grid. This is an example of robot localization over time, and is a key capability of a mobile robot. However, it will have to wait until we can do efficient inference.

1.10 Maximum a Posteriori Estimates

Finally, another well known inference problem is the **maximum a posteriori** or MAP estimate: given the values of some variables $\mathcal{Z}$, what is the most probable joint assignment to a subset $\mathcal{X}$ of the other variables? In this case the variables are partitioned into three sets: the variables of interest $\mathcal{X}$, the nuisance variables $\mathcal{Y}$, and the observed variables $\mathcal{Z}$. We have

$$P(\mathcal{X}|\mathcal{Z} = \mathfrak{z}) = \sum_{\mathfrak{y}} P(\mathcal{X}, \mathcal{Y} = \mathfrak{y}|\mathcal{Z} = \mathfrak{z}) \propto \sum_{\mathfrak{y}} P(\mathcal{X}, \mathcal{Y} = \mathfrak{y}, \mathcal{Z} = \mathfrak{z}). \tag{5}$$

Finding a MAP estimate is more expensive than finding the MPE, as in addition to enumerating all possible combinations of $\mathcal{X}$ and $\mathcal{Y}$ values, we now need to calculate a possibly large number of sums, each exponential in the size of $\mathcal{Y}$. In addition, the number of sums is exponential in the size of $\mathcal{X}$. Below we will see that while we can still exploit the Bayes net structure, MAP estimates are fundamentally more expensive even in that case.

Exercise

Prove that we need only a single normalization constant in Equation 5. This is easiest using the standard form of Bayes' rule, Equation 2.

Summary

We briefly summarize what we learned in this section:

1. A Grid world is a more realistic robotics example.
2. Conditional probability distributions allow for modeling sensors.
3. We can compute a joint probability distribution, and marginal and conditionals from it.
4. Bayes nets allow us to encode more general joint probability distributions over many variables.
5. Ancestral sampling is a technique to simulate from any Bayes net.
6. Dynamic Bayes nets unroll time and can be used to simulate robots over time.
7. Bayes' rule allows us to infer knowledge about a state from a given observation.

8. Inference in Bayes nets is a simple matter of enumeration, but this can be expensive.
9. The maximum probable explanation singles out one estimate.
10. Marginalizing over some variables leads to MAP inference.