Contents

1 Probability and Bayes Nets

Motivation

Bayes nets give us a way to simulate robots. We will view simulation as drawing samples from a structured probability distribution. The probability distribution can be viewed as a stochastic world model, by describing how sensors reflect state, and how actions affect the state.

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. The robot has a sensor to figure out where it is, and it's action space is limited to moving left, right, up, or down in the grid.

The random variables we use below can also be used to describe the state of a real robot, and the data structures we build below can be used to control and plan for a real robot. The discrete and coarse nature of our representation will manifest itself in being able to make only probabilistic statements about the robot's state, which is called inference. In this section we will do some of that, but will mostly be content with the modeling part.

Below we take a **Bayesian view of probability**, rather than a frequentist one. This means that we see probabilities as describing our knowledge about events, rather than tallying up frequencies by which they occur. Think of the weather-person talking about the probability of rain tomorrow. Probabilities viewed this way can be used to describe **prior knowledge** about the state of the world, how sensors work, and how actions affect the state of an agent and the world.

As we introduce concepts below, we will also introduce a graphical way to represent these concepts, and as such gradually develop the general notion of a Bayes net, which is a graphical model to describe complex probability distributions.

1.1 Discrete Distribution

Figure 1: Graphical representation of a single random variable.

A **discrete probability distribution** is the probability distribution of a random variable X, taking on one of K discrete values. For example, we can model the action space of the robot with a random variable A, taking on values $a \in \{Left, Right, Up, Down\}$. Note that we denote random variables with a capitalized symbol, e.g., A for action, but when in a formula we talk about the *value* of a random variable, we use a lowercase a.

We can represent the parameters of a discrete probability distribution as a vector of probabilities $p_i \triangleq P(X = x_i)$, called the **probability mass function** or PMF. Probability mass functions obey two basic axioms,

$$p_i \geq 0$$
$$\sum p_i = 1$$

i.e., the probability p_i of an outcome x_i is non-negative, and the probabilities p_i of all outcomes have to sum up to 1.

Graphically, we can represent a PMF over a single random variable X as a single node, with the label X, as shown in Figure 1. By convention we use a circular node to represent random variables.

Example

Below is an example PMF describing the distribution over actions for our example robot. This encodes the knowledge that the robot seems more likely to move the right.

a_i	$P(A = a_i)$
Left	0.2
Right	0.6
Up	0.1
Down	0.1

Table 1: Probability mass function.

Grid-world example

To describe the location or state S of the robot, let us use 10×10 grid, in which case there are 100 possible outcomes. A 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. Some examples for such a PMF include: uniform, a single particular starting position, more probable to be the on left, etc...

An example is shown in Figure 2, which models that given no other information, we expect to find the robot in one of 10 different possible locations with equal probability.

1.2 Sampling

Given a distribution, we can draw a sample from it. To do so, we use an algorithm called **inverse transform sampling**. For this we need to first assign an order to the outcomes,, which we can do

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

Table 2: A PMF describing where a robot might be in a grid.

by adopting order associated with the (arbitrary) integer indices by which we enumerate outcomes, i.e.: $x_i < x_j$ if $i < j$. Given this order, we can the then compute the **cumulative distribution function** or CDF, which is the probability associated with the subset of outcomes with index less than or equal to a given index i:

$$F(x_i) = P(X \leq x) = \sum_{j<i} P(X = x_i) = \sum_{j<i} p_j$$

Given this, the inverse transform algorithm is simple: generate random number $0 \leq u \leq 1$, then return x_i such that $i$is the largest number with $F(x_i) \leq u$.

Example

a_i	$P(A = a_i)$	$F(a_i)$
Left	0.2	0.2
Right	0.6	0.8
Up	0.1	0.9
Down	0.1	1.0

Table 3: Cumulative distribution function.

In Table 3, we have shown the CDF for the robot action, using the order from top to bottom. To sample, a random value

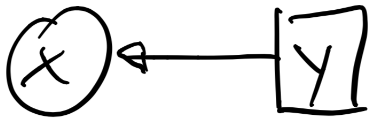
$u = 0.6$ would yield a sample $X = Right$.

Exercises

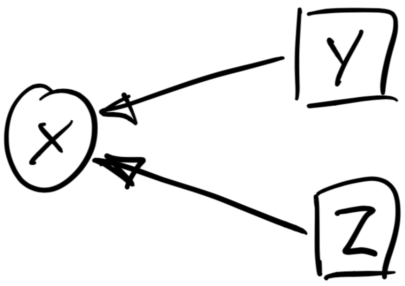
1. Implement inverse transform sampling for the CDF in Table 3, and verify your sampler produces a histogram that approximates the desired PMF.

2. Calculate the CDF for the grid world example in Figure 2.

1.3 Conditional Distribution



(a) $P(X|Y)$



(b) $P(X|Y,Z)$

Figure 2: Conditional probability distributions. Above X is a random variable, but Y and Z are known parameters, indicated by the square box.

The probability mass function of a single variable X could be parameterized by a parameter Y, whose value we will assume as known for now. This corresponds to the notion of a conditional probability, which we write as

$$P(X|Y=y).$$

Note that given a particular value of Y, this is just a distribution over X, with parameters given by a PMF, as before. But, because Y can take on several values, we now need a **conditional probability table** or CPT to exhaustively describe our knowledge.

Graphically, we represent a known parameter Y as a square node within a graph, and a random variable governed by a conditional probability as having an incoming edge from the parameter, as shown in Figure 2a. Note that it is possible to have multiple parameters, e.g., a conditional probability $P(X|Y,Z)$ is shown in Figure 2b.

We can sample from a conditional distribution $p(X|Y)$ by selecting the appropriate PMF, depending on the value of Y, and proceeding as before using the inverse transform sampling method.

Example

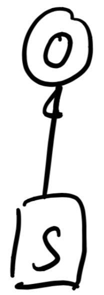
a_i	$P(A=a_i \mid T=Left)$	$P(A=a_i \mid T=Right)$
Left	0.6	0.2
Right	0.2	0.6
Up	0.1	0.1
Down	0.1	0.1

Table 4: Conditional probability table (CPT).

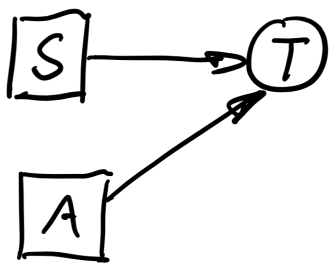
Table 4 gives an example where we have a variable T to differentiate between robots tending to the left, or tending to the right.

1.4 Modeling the World

Conditional probability distributions are a great way to represent knowledge about the world in robotics. In particular, we will use them to model sensors, as well as how we can affect the state of the robot by actions.



(a) Sensor Model $P(O|S)$



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

Figure 3: Conditional distributions to model sensing and acting.

Assuming a robot has a single sensor, or we observe the robot from a different vantage point, 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 3a. 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.

Conditional probability tables 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 give the CPT 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:

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

We can model the effects of actions in a similar manner, by a conditional probability distribution $P(T|S = s, A = a)$ on th next state T , given the value s of the current state S, and the value a of the action A, bit viewed as parameters. 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 associated CPT.

Exercise

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

Exercise

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

1.5 Joint Distribution

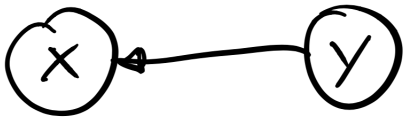


Figure 4: Joint probability distribution on two variables X and Y.

If we consider the parameter Y in a conditional probability distribution not as a known parameter but as a random variable itself, with probability distribution $P(Y)$, we obtain a **joint probability distribution** over the pair of variables X and Y, and its PMF is given by the **chain rule**:

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

Graphically, we have the graph fragment shown in Figure 4.

To sample from a joint distribution $P(X, Y)$ we can take the graph as a guide: 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 $P(x|y)$.

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:

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

Exercises

1. Viewing the integer coordinates i and j as separate random variables, use the distribution $P(i, j) = P(S)$ from Figure 2 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 Figure 2.

1.6 Bayes' Rule

Given the formulas we discussed above, we can 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 discuss in Section 1.7.

Bayes' rule allows 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**,

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

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 weigthed 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 Figure 2 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?

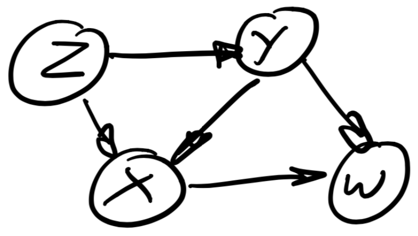


Figure 5: A Bayes net representing a more general joint distribution over the random variables W,X,Y, and Z.

1.7 Bayes Nets

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 5, and it is simply a graphical representation of which random variables's CPT depend on which other variables. In this case, the joint distribution can be read off as

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

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 5: Number of entries in each CPT for the Bayes net of Figure 5, assuming all variables have 10 outcomes.

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 5. In Table 5 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 X and Y. Hence, the total number of parameters we need is only $1,899$, which is way less than $9,999$.

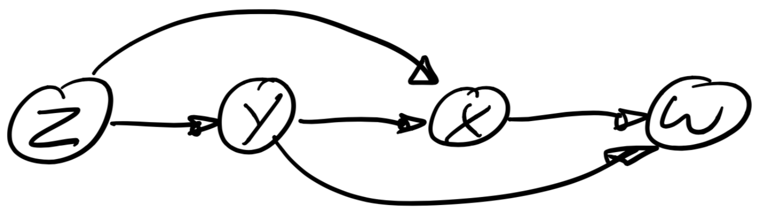


Figure 6: The topological sort of the Bayes net in Figure 5.

1.8 Ancestral Sampling

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 6, which shows a topological sort of the Bayes net in Figure 5. 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.

1.9 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.

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 7. We recognize subgraphs modeling the sensor behavior, and the effects of actions, at each time step. 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
$$P(S_k|S_k = s_k, A_k = a_k)$$
.
5. Increase k and return to step 2.

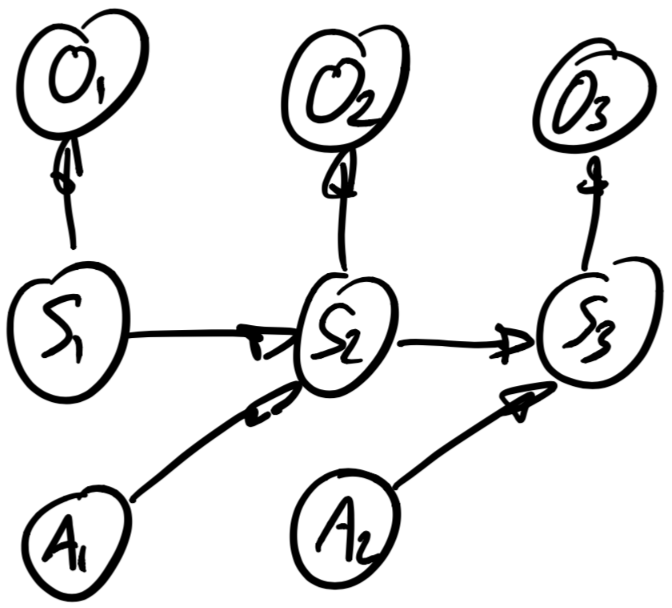


Figure 7: A Bayes net modeling our robot example.

Exercise

Simulate two different realizations from the dynamic Bayes net in Figure 7.

1.10 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 $\mathcal{Z}$.

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 5, 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 6.

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 $W=3$

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 6: 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.

and $X = 6$. Note that to compute the MPE, we need not bother with normalizing: we can simply find the maximum entry in the unnormalized posterior values.

In both these inference problems, 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 6 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.

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 7, 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.

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, exponential in the size of $\mathcal{Y}$. The number of sums is exponential in the number 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.

Exercises

1. Show that in the example from Figure 5, 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 7, 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.
4. 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. Discrete distributions model the outcome of a single categorical random variable.
2. Inverse transform sampling allows us to simulate a single variable.
3. Conditional probability distributions allow for dependence on other variables.
4. Models for sensing and acting can be built using parametric conditional distributions.
5. We can compute a joint probability distribution, and marginal and conditionals from it.
6. Bayes' rule allows us to infer knowledge about a state from a given observation.
7. Bayes nets allow us to encode more general joint probability distributions over many variables.
8. Ancestral sampling is a technique to simulate from any Bayes net.
9. Dynamic Bayes nets unroll time and can be used to simulate robots over time.
10. Inference in Bayes nets is a simple matter of enumeration, but this can be expensive.