Assignment 5—Bayes Nets

NOTE: If you're reading this with github's Markdown renderer, you're going to see a lot of $'s all over the place because it can't handle LaTeX. You should read the README.pdf, instead.

Context

Up until now, the reasoning that we've done has relied on us being *certain* about the state of the world—we *knew* that Willie wasn't on stage, or that Zuohao was wearing a yellow hat, or so on. Occasionally, we ran across pieces of information that weren't explicitly, but we've assumed a closed world and the law of the excluded middle, so life hasn't been *too* complicated for us.

Unfortunately, many situations (robotics, medicine, criminal justice, economics, etc.) don't afford us with perfect knowledge of the world, nor are we granted the liberty of assuming state just because we're ignorant of some particulars. We need to be able to reason about situations where we *don't* have complete information, just a few estimates for how certain pieces fit together.

When we want to reason about uncertainty we can turn to bayesian networks as our tool of choice. Bayes nets, in a nutshell, let us use a bit of math to reason about the probability of a particular hypothesis being true (or false) given a set of evidence. Hypotheses are states of the world that we want to reason about, and they include things like, "does this patient have HIV?," "is there a clear path across this room?," and the infamous "am I being robbed or is this an earthquake?" Evidence is world state that we know to be one way or another, such as "I know there are no chairs here," "the patient has this set of symptoms," and so on.

We reason about hypotheses (whether or not they hold) given evidence, and for bayes nets in particular this manifests as questions like, "what is the probability that is isn't raining, given that the grass is wet and it's cloudy?"

Your Task

You've been given a file, `student_code.py`, which consists of the lonely little function definition for `ask`. Right now, `ask` doesn't do anything. Your job will be to implement it.

The `ask` function needs to return the probability of a hypothesis given some evidence: P(H|E).
The function takes four arguments: * `var` is the name of the hyptothesis variable. * `value` indicates whether the hypothesis being asked about is `True` or `False`. * `evidence` is the set of variables known to be `True` or `False`. * `bn` is the `BayesNet` (in the provided module) pertaining to the problem.

To calculate P(H|E), `ask` should calculate and return $\frac{P(H,E)}{\alpha}$, where:

- $P(H, E)$ is the joint probability of the hypothesis (`var` = `value`) and the evidence (`evidence`), and
- α is the normalization constant (the joint probability of ¬hypothesis and the evidence).

See below for more details on calculate $P(H, E)$ and α.

Testing

The provided tests use the Burglary model discussed in class. You should be able to easily create more test cases with this model. Additionally, you are encouaged to create your own instance of BayesNet that models a different domain (e.g., cavities, traffic, or exam grades).

Notes

On Joint Probability

To calculate the joint probability, we can break things down into terms that we can just lookup in the BayesNet. For example, the Burglary model does not represent $P(b, e, a)$, but it does have $P(b)$, $P(e)$, and $P(a|b, e)$.
We can handle this recursively by recognizing the following: $P(b, e, a) = P(b)P(e, a|b)$ (This is called the Chain Rule). Similarly, we know that $P(e, a|b) = P(e|b)P(a|b, e)$.

When calculating the joint probability, we need to include all the variables that may influence the final result. This includes all the variables that are parents of the variables in the call to `ask`. For example, in calculating P(b,j,m), we need to include A and E (whose values we do not know; more on that in a bit). In fact, we can add in all the variables in the BayesNet, as the extra unknown variables will not affect the final result. They only add a little extra computation.

To implement this recursion you may want to introduce a new function. That function takes the list of variables in the joint probability and the collection of all known variable values. For example, for $P(e, a|b)$, the function takes the list `[E,A]` as the list of variables. The known variable values can be represented with the dictionary `{'E':True, 'A': True, 'B':True}`.

This new function should handle the following conditions: * Recursion is done if there are no more variables in the list. * The next variable in the list has a known value (it is in the evidence). * The value of the next variable is not known.

In the case where the next variable has a known value, lookup the probability in the CPT using the function `probability` on the BayesNode. Then recurse on the rest of the variables.

If the value of the next variable is not known, we need to compute the sum of the joint probabilities when the unknown is True or False. In other words, when trying to find out $P(B, e, a)$ given unknown B and known e and a, $P(B, e, a) = P(b, e, a) + P(\neg b, e, a) = P(b)P(e, a|b) + P(\neg b)P(e, a|\neg b)$. For each of the two possibilities, lookup the probability in the CPT and recurse. Note that when recursing, we have now defined a value for the first variable, and this value needs to be included in the recursive call.

In calculating the joint probability, it is best to process the variables orderly. If the list of variables is ordered such that the parent of any variable precedes its children, then when processing the child, we will already know that value. For example, in calculating $P(a, B, e)$, if we order the variables such that we calculate $P(B, e, a)$, then when processing `a`, we will have specified a value for `B` and can thus lookup the value of $P(a|b, e)$.

To get the list of variables in order and to get the whole list of variables, use `BayesNet.variables`.

Variables and known values

In code, we represent the values of variables as a dictionary keyed on the *name* of the variable. The value of the dictionary entry is either `True` or `False`.

An example:

```
{
  'Alarm': True, 'Burglar': False, 'Earthquake': False,
  'JohnCalls': False, 'MaryCalls': False
}
```

This representation is used for the `evidence` argument in the call to `ask`. In making your recurive function call, you will want to take the given evidence and update it (in the 3rd case described above). You may notice that plenty of variables are going to be left out, and this is okay, as Bayes rule works just fine with unknowns (since that's the point...).

On The Normalization Constant

The normalization constant is the sum of: (1) the joint probability of the evidence and hypothesis and (2) the joint probability of *not* the hypothesis and the evidence. For example, the normalization constant α for $P(a|b)$ is:

$$\alpha = P(a, b) + P(\neg a, b)$$

On the BayesNet Module

The most important field for a `BayesNet` (the class in the BayesNet module that represents... a bayes net) is the `variables` field, which is an iterator of all of the variables in the net.

Each variable in a `BayesNet` is an instance of `BayesNode`, of which the most important fields are `name` and `evidence`. You will also need to use the `probability` function to lookup probabilities in the CPT.