Conditioning - exercises

Exercise 1: Fair coins and biased coins

a)

I flip a fair coin. What is the probability that it lands heads?

```
var model = function() {
  // Your code here
}
var log_prob = Infer({method:'enumerate'}, model).score('H')
Math.exp(log_prob)
```

run ▼

b)

I also have a biased coin, with $P(\text{heads}) = 0.9$P(heads)=0.9. I hand you one of the coins (either biased or fair) without telling you which. You flip it three times.

Given that first two coin flips landed on heads, what is the posterior distribution for the next flip?

```
var model = function() {
  // Your code here
}
viz(Infer({method:'enumerate'}, model))
```

run ▼

c)

Given that all three flips landed on heads, what is the probability that the coin was biased?

d)

Given that the first two flips were different, what is the probability that the third flip will be heads?

Exercise 2: Conditioning and Intervention

In the earlier Medical Diagnosis (/chapters/02-generative-models.html#example-causal-models-in-medical-diagnosis) section we suggested understanding the patterns of symptoms for a particular disease by changing the prior probability of the disease such that it is always true (also called the *do* operator).

```
var lungCancer = flip(0.01);

var cold = flip(0.2);

var cough = (

  (cold && flip(0.5)) ||

  (lungCancer && flip(0.3))

)

cough;
```

run ▼

a)

For this example, does intervening on the program in this way (e.g. by setting the value of `lungCancer`) have the same effect as *conditioning* on the disease being true? What about the causal dependency makes this case?

b)

Why would intervening have a different effect than conditioning for more general hypotheticals? Construct an example where they differ. Then translate this into a WebPPL model and show that manipulating the prior gives different answers than manipulating the observation. *Hint:* think about the effect that intervening vs. conditioning on a variable that has a **causal parent** would have on that parent.

```
x
```

run ▼

Exercise 3: Computing marginals

Use the rules for computing probabilities to compute the marginal distribution on return values from these programs by hand (use `viz()` to check your answers):

a)

```
Infer({method: "enumerate"}, function() {
  var a = flip();
  var b = flip();
  condition(a || b);
  return a;
})
```

run ▼

b)

```
var smilesModel = function() {
  var nice = mem(function(person) {return flip(.7)});
  var smiles = function(person) {return nice(person) ? flip(.8) : flip(.5);}
  condition(smiles('alice') && smiles('bob') && smiles('alice'));
  return nice('alice');
}

Infer({method: "enumerate"}, smilesModel)
```

run ▼

Exercise 4: Extending the smiles model

a)

Describe (using ordinary English) what the second WebPPL program, `smilesModel` above means.

b)

Extend `smilesModel` to create a version of the model that also captures these two intuitions:

1. people are more likely to smile if they want something and
2. *nice* people are less likely to want something.

Note: Do not lose the fact that niceness is also a risk factor for smiling.

Hint: Which variables change at different times for the same person? Which values *depend* on other values?

```

var extendedSmilesModel = function() {

  var nice = mem(function(person) {return flip(.7)});


  ...


  var smiles = function(person, ...) {

    return nice(person) ? flip(.8) : flip(.5);

  }


  return smiles('alice')

}

```

```
Infer({method: "enumerate"}, extendedSmilesModel)
```

run ▼

c)

Suppose you've seen Bob five times this week and each time, he was not smiling. But today, you see Bob and he *is* smiling. Use this `extendedSmilesModel` model to compute the posterior belief that Bob wants something from you today.

Hint: How will you represent the same person (Bob) smiling *multiple times*? What features of Bob will stay the same each time he smiles (or doesn't) and what features will change?

In your answer, show the WebPPL inference and a histogram of the answers – in what ways do these answers make intuitive sense or fail to?

```
var extendedSmilesModel = function() {
  // copy your code frome above

  // make the appropriate observations

  // return the appropriate query
  return ...;
}


Infer({method: "enumerate"}, extendedSmilesModel)
```

run ▼

Exercise 5: Sprinklers, Rain and mem

a)

I have a particularly bad model of the sprinkler in my garden. It is supposed to water my grass every morning, but is turns on only half the time (at random, as far as I can tell). Fortunately, I live in a city where it also rains 30% of days.

One day I check my lawn and see that it is wet, meaning that either it rained that morning or my sprinkler turned on (or both).

Answer the following questions, either using the Rules of Probability or by writing your own sprinkler model in webppl.

- What is the probability that it rained?
- What is the probability that my sprinkler turned on?

run ▼

b)

My neighbour Kelsey, who has the same kind of sprinkler, tells me that her lawn was also wet that same morning. What is the new posterior probability that it rained?

run ▼

c)

To investigate further we poll a selection of our friends who live nearby, and ask if their grass was wet this morning. Kevin and Manu and Josh, each with the same sprinkler, all agree that their lawns were wet too. Using `mem`, write a model to reason about arbitrary numbers of people, and then use it to find the new probability that it rained.

run ▼

Exercise 6: Casino game

Consider the following game. A machine randomly gives Bob a letter of the word "game"; it gives a, e (the vowels) with probability 0.45 each and the remaining letters (the consonants g, m) with probability 0.05 each. The probability that Bob wins depends on which letter he got. Letting hh denote the letter and letting $Q(h)$Q(h) denote the numeric position of that letter in the word "game" (e.g., $Q(\mathrm{g}) = 1, Q(\mathrm{a}) = 2$Q(g)=1,Q(a)=2, and so on), the probability of winning is $1/Q(h)^2$1/Q(h)2.

Suppose that we observe Bob winning but we don't know what letter he got. How can we use the observation that he won to update our beliefs about which letter he got? Let's express this formally. Before we begin, a bit of terminology: the set of letters that Bob could have gotten, $\{g, a, m, e\}${g,a,m,e}, is called the *hypothesis space* – it's our set of hypotheses about the letter.

a)

In English, what does the posterior probability $p(h \mid \text{win})$p(h | win) represent?

b)

Manually compute $p(h \mid \text{win})$p(h | win) for each hypothesis. Remember to normalize — make sure that summing all your $p(h \mid \text{win})$p(h | win) values gives you 1.

h**h**	$p(h)$**p(h)**	$p(\text{win} \mid h)$**p(win ∣ h)**	$p(h \mid \text{win})$**p(h ∣ win)**
g	0.05		
a	0.45		
m	0.05		
e	0.45		

c)

Now, we're going to write this model in WebPPL using `Infer`. Here is some starter code for you:

```
// define some variables and utility functions

var checkVowel = function(letter) {return _.includes(['a', 'e', 'i', 'o', 'u'], letter);}

var letterVals = ['g', 'a', 'm', 'e'];

var letterProbs = map(function(letter) {return checkVowel(letter) ? 0.45 : 0.05;}, letterVals

var letters = Categorical({vs: letterVals, ps: letterProbs})



// Compute p(h | win)

var distribution = Infer({method: 'enumerate'}, function() {

  var letter = sample(letters);

  var position = letterVals.indexOf(letter) + 1;
  var winProb = 1 / Math.pow(position, 2);

  condition(...)

  return ...

});

viz.auto(distribution);
```

run ▼

Fill in the `...`’s in the code to compute $p(h \mid \text{win})$p(h | win). Include a screenshot of the resulting graph. What letter has the highest posterior probability? In English, what does it mean that this letter has the highest posterior? It might be interesting to comment out the `condition` statement so you can compare visually the prior (no `condition` statement) to the posterior (with `condition`).

Make sure that your WebPPL answers and hand-computed answers agree – note that this demonstrates the equivalence between the program view of conditional probability and the distributional view.

d)

Which is higher, $p(\text{vowel} \mid \text{win})$p(vowel | win) or $p(\text{consonant} \mid \text{win})$p(consonant | win)? Answer this using the WebPPL code you wrote *Hint:* use the `checkVowel` function.

```

// define some variables and utility functions

var checkVowel = function(letter) {return _.includes(['a', 'e', 'i', 'o', 'u'], letter);}

var letterVals = ['g', 'a', 'm', 'e'];

var letterProbs = map(function(letter) {return checkVowel(letter) ? 0.45 : 0.05;}, letterVals

var letters = Categorical({vs: letterVals, ps: letterProbs})


// Compute p(h | win)

var distribution = Infer({method: 'enumerate'}, function() {
  var letter = sample(letters);
  var position = letterVals.indexOf(letter) + 1;
  var winProb = 1 / Math.pow(position, 2);
  condition(...)
  return ...
});
viz.auto(distribution);
```

run ▼

e)

What difference do you see between your code and the mathematical notation? What are the advantages and disadvantages of each? Which do you prefer?