Towards programming inference

Today we will continue designing code for probabilistic inference on finite sets. You will follow up outside of class and bring solutions in one week, on Monday, September 22. Please bring your computer to class for show and tell.

Two notes:

- The problems are intended for you to work on together, even outside of class.
- I suspect that everybody learns more and has more fun if we can find a way to discuss the problems on Piazza. But I’m not sure how to manage those discussions so as to avoid spoilers. I may ask you to try to put spoilers in notes labeled SPOILER in the title.

Review of the big picture

We’re treating “probability distribution” as an abstract data type. And we’re designing functions that work with distributions. To keep track of what’s in a probability distribution, we’re using parametric polymorphism (type parameters). So assume you have an abstract type constructor $P$, such that $P$ a is the type of probability distributions over values of type $a$.

If $D$ is the type of a single die and $Col$ is either $\text{LEFT}$ or $\text{RIGHT}$, here are some example types:

<table>
<thead>
<tr>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P D$</td>
<td>distribution of a single die (in ML, $d p$)</td>
</tr>
<tr>
<td>$P \text{Int}$</td>
<td>distribution containing, e.g., the result of rolling one die, or the total of some collection of dice, or really any probabilistic computation that results in a number (in ML, $\text{int } p$)</td>
</tr>
<tr>
<td>$P (D, D)$</td>
<td>distribution of two dice (in ML, $(d \times d) p$)</td>
</tr>
<tr>
<td>$P \text{Col}$</td>
<td>distribution of a single mark on the tally sheet, either $\text{LEFT}$ or $\text{RIGHT}$ ($\text{col } p$)</td>
</tr>
<tr>
<td>$P [\text{Col}]$</td>
<td>distribution of a list of marks on the tally sheet ($\text{col list } p$)</td>
</tr>
<tr>
<td>$P \text{Int}$</td>
<td>distribution of number of marks on right side of tally sheet ($\text{int } p$)</td>
</tr>
</tbody>
</table>

Using distribution types

To solve simple problems, people usually start with values, use values to form distributions, use distributions to make other distributions, and eventually ask questions about distributions that get back to values. When solving more complicated problems, we may go back and forth between distributions and values multiple times.

The best results from last time

Here are the most durable results I saw from our last class. (I have changed the names of functions.)

- From the side board,

  ```haskell
equally :: [a] -> P a -- Haskell
val equally : 'a list -> 'a p (* SML *)
```

  Function `equally xs` returns a distribution in which each element of `xs` is equally likely to occur. Or to put it another way, the probability of any value occurring in the distribution `equally xs` is proportional to the number of times that value occurs in `xs`.

- From the front board,

  ```haskell
sample :: Entropy -> P a -> a -- Haskell
sample : 'a p -> 'a (* SML *)
```

  This is the classic function that allows you to sample a value from a distribution. In a pure language, or in a semantics, you need to account for a source of randomness, which here I have called `Entropy`. In an impure language you can just consume randomness as a side effect.

These two functions are related by the **Law of Large Numbers**: suppose you’re given a distribution `dist` of type $P$ a and you form a list of length $N$ by sampling from `dist` $N$ times using distinct sources of entropy. Then as $N$ gets large, distribution `equally as` becomes a better and better approximation of the original `dist`.

Interpretations of a distribution

An abstract probability distribution may have multiple interpretations, ranging from the extremely mathematical territory of
measure theory to the extremely operational territory of simulating thrown dice. Multiple interpretations are valid, and as a language designer, you can choose an interpretation that suits your purpose. But for understanding, it is worth trying to discover as many interpretations as possible.

The design task in class, Part I: General probability distributions

Questions about creating distributions:

1. What functions can you think of, in addition to equally, that will introduce new probability distributions?
2. What functions can you think of that will take existing distributions and transform them into new distributions?
3. What functions can you think of that will combine two or more distributions into a single distribution?
4. What functions can you think of that will enable you to combine dependent distributions? For example: draw a die from the bag, then throw that die once. What is the resulting distribution over (die, number) pairs? How do you compute it?

Question about once you have a distribution, then what?

5. Given a distribution, what do you think it ought to be reasonable to do with it? (Assume that there are only finitely many values that have nonzero probability.) Can one do anything besides sample?

The design task in class, Part II: Solving probability problems

As a test of your ideas about abstract probability distributions, which of the following problems will your functions from Part I enable you to solve?

By Monday, September 22, I expect that you will have written code to solve at least one question in each category.

Simple distribution and probability questions

A. What distribution of integers results from throwing a single d6? What about a d12?
B. Given that you have a d6 and a d12, what’s the probability of throwing a total of 11?
C. You’re given a \( P \) (distribution over dice) representing the ability to draw one die at random from a bag. You draw two dice. (Assume you are drawing dice “with replacement”, so that you can draw two by doing the exact same probabilistic draw twice.)

• What’s the probability of drawing a d6 and a d12?
• What’s the probability that exactly one of the two dice is a d20?

D. You draw two dice, throw them, and total the numbers. What’s the (“joint”) probability of drawing a d6 and a d12 and throwing 11?
E. You draw two dice and throw them. The numbers total 11. Given that you know the total of 11, what’s the (“conditional”) probability that you drew a d6 and a d12?

Tally-sheet questions

In all these questions, you throw a tally sheet (draw two dice, throw 30 times, mark left column for totals of at most 7 and right column for totals of at least 8).

F. What’s the probability that you have drawn two d4’s and put 3 marks in the right column of the tally sheet?
G. On average, how many marks can you expect to put in the right column of the tally sheet?

Gambling questions

Probability was invented by gamblers for gamblers. Let me describe the tally-sheet game:

The tally-sheet game has two players: the thrower and the guesser.

• The thrower creates a tally sheet by drawing two dice and throwing them 30 times using the procedure you know.
• The guesser sees the tally sheet but not the dice and has three chances to try to guess the dice.

It’s not gambling unless you play for money. Here are the payouts:

• If the guesser gets the right dice on the first try, the guesser wins one dollar ($1.00) from the thrower.
• If the guesser gets the right dice on the second try, the guesser wins fifty cents ($0.50) from the thrower.
• If the guesser gets the right dice on the third try, the guesser wins a quarter ($0.25) from the thrower.
• If the guesser fails to get the right dice in three tries, the guesser loses and pays a dime ($0.10) to the thrower.
Write code to answer the following questions about the tally-sheet game.¹

H. Who is likely to win money: the guesser or the thrower?

I. You have your choice of playing guesser or thrower for a series of 1000 games. You choose whichever is more likely to win. How much do you expect to win?

J. You’re playing thrower, and I ask if you’d be willing to change the rules so that instead of throwing 30 times, you throw 50 times. Assuming we’re going to play 1000 games, what’s the minimum amount you should insist on charging me for this privilege?

K. As guesser, I would like the option to change the rules so that the right column tallies throws of 10 or more instead of 8 or more. Assuming I’m going to play 1000 games as guesser, what’s the most I should be willing to pay for the privilege of exercising this option?

L. The split between 7 and 8 is rather arbitrary, and it makes life hard for the guesser. Suppose we change the split to some other point on the number line. What split maximizes the expected return for the guesser?

The coding task

In the language of your choice, in teams of your choice, implement probability distributions in a way that enables you to run code that answers the questions above. Answer at least one question from each group. Bring your solutions to class for Show and Tell on Monday, September 22.

Here is some advice:

- Prefer a language that has a static, polymorphic type system. Standard ML (from COMP 105) or Haskell (from COMP 150FP) are both entirely adequate to this task. Other reasonable choices include F#, OCaml, Scala, and (possibly) Java with generics. You could even consider C++ with templates, especially if you’re a good C++ programmer and you use libgc.

- No matter what language you go with, choose an immutable representation of probability distributions.

- Keep your eye on the types of intermediate results—you have got to keep the state space small, or your computations won’t terminate in your lifetime. The size of the state space is simply the number of values in a distribution that have nonzero probability.

- Be careful to keep the size of your representation proportional to the size of the state space. If you’re very naive

---

¹In all these questions, please assume that the cost of throwing dice and the cost of computing answers are both zero. In other words, the questions should consider only the payouts from the game; all actions in the real world are free.