

Computational Probability and Inference

L^AT_EXdocument by Colin Leach, from an HTML original by MIT course staff
September 2016

Abstract

This is a set of notes for the online course ‘6.008.1x Computational Probability and Inference’ given by the staff of MIT.

Contents

1	Introduction to Probability	3
1.1	Introduction	3
1.2	A First Look at Probability	3
1.3	Probability and the Art of Modeling Uncertainty	5
2	Probability Spaces and Events	5
2.1	Two Ingredients to Modeling Uncertainty	5
2.2	Probability Spaces	6
2.3	Table Representation	7
2.4	More on Sample Spaces	7
2.5	Probabilities with Events	8
2.6	Events as Sets	8
2.7	Code for Dealing with Sets in Python	8
2.8	Probabilities with Events and Code	8
3	Random Variables	9
3.1	A First Look at Random Variables	9
3.2	Random Variables	9
3.3	Random Variables Notation and Terminology	11
4	Jointly Distributed Random Variables	11
4.1	Relating Two Random Variables	11
4.2	Representing a Joint Probability Table in Code	13
4.3	Marginalization	15
4.4	Marginalization for Many Random Variables	17
4.5	Conditioning for Random Variables	18
4.6	Moving Toward a More General Story for Conditioning	19
5	Conditioning on Events	19
5.1	Conditioning on Events Intro	19
5.2	The Product Rule for Events	19
5.3	Bayes’ Theorem for Events	19
5.4	Practice Problem: Bayes’ Theorem and Total Probability	19
5.5	Take-Away Lessons:	21
6	Inference with Bayes’ Theorem for Random Variables	21
6.1	The Product Rule for Random Variables (Also Called the Chain Rule)	21
6.2	Bayes’ Rule for Random Variables (Also Called Bayes’ Theorem for Random Variables	23
6.3	Bayes’ Theorem for Random Variables: A Computational View	23
6.4	Maximum A Posteriori (MAP) Estimation	24

7	Independence Structure	25
7.1	Independent Events	25

1 Introduction to Probability

1.1 Introduction

Probabilities appear in everyday life and feed into how we make decisions. For example:

The weather forecast might say that “tomorrow there is a 70% chance of rain”. This 70% chance of rain is a probability, and if it is sufficiently high, then we may want to bring an umbrella when we go outdoors.

We could predict that the probability of car traffic is higher during rush hour than otherwise, so if we don’t want to be stuck in traffic while driving, we should avoid driving during rush hour.

We aim to build computer programs that can reason with probabilities.

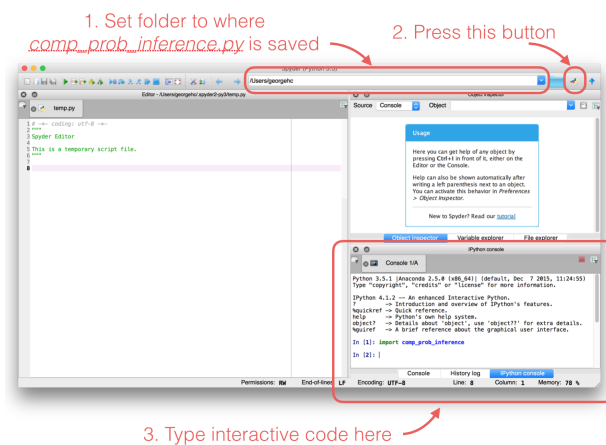
1.2 A First Look at Probability

Perhaps the simplest example of probability is flipping a fair coin for which we say that the probability of heads is $1/2$ and, similarly, the probability of tails is also $1/2$. (Don’t worry, we’ll see much more exciting problems soon!) What do we mean when we say that the probability of heads is $1/2$?

The basic idea is that if we repeat this experiment of flipping a coin a huge number of times, say n , then the number of heads we should see should be close to $n/2$ as we increase n . While you could certainly try this out in real life by flipping a coin, say, 100,000 times, doing this would be disastrously tedious. Let’s simulate these flips in Python instead.

Simulating Coin Flips Follow along in an IPython prompt within Spyder.

We have provided a package `comp_prob_inference.py`, which you should save to your computer. Within Spyder, do the following:



Let’s start by importing the package `comp_prob_inference`:

```
> import comp_prob_inference
```

To simulate flipping a fair coin, enter:

```
> comp_prob_inference.flip_fair_coin()
```

You should get either ‘heads’ or ‘tails’. Try re-running the above line a few times. You should see that the coin flip results are random.

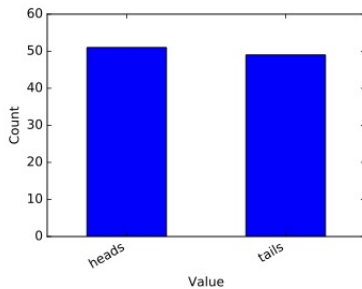
To flip the fair coin 100 times, enter:

```
> flips = comp_prob_inference.flip_fair_coins(100)
```

Let's plot how many times we see the two possible outcomes in the same bar graph, called a histogram:

```
> comp_prob_inference.plot_discrete_histogram(flips)
```

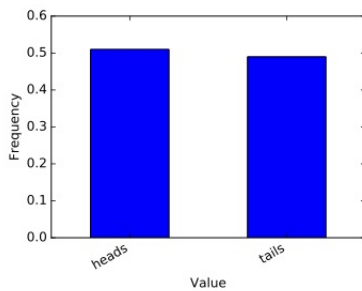
For example, we get the following plot:



Often what we will care about in this course is the fraction (also called the frequency of times an outcome happens). To plot the fraction of times heads or tails occurred, we again use the `plot_discrete_histogram` function but now add the keyword argument `frequency=True`:

```
> comp_prob_inference.plot_discrete_histogram(flips, frequency=True)
```

Doing so, we get the following plot:



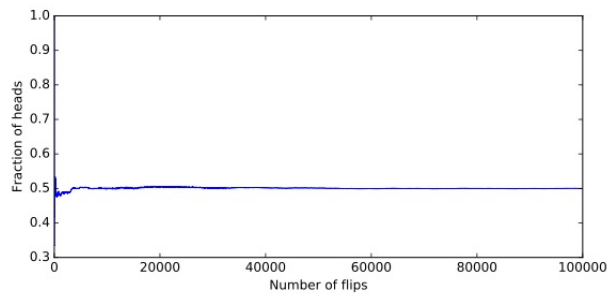
Next, let's plot the fraction of heads as a function of the number of flips (going up to 100,000 flips).

```
n = 100000
heads_so_far = 0
fraction_of_heads = []
for i in range(n):
    if comp_prob_inference.flip_fair_coin() == 'heads':
        heads_so_far += 1
    fraction_of_heads.append(heads_so_far / (i+1))
```

Note that `fraction_of_heads[i]` tells us what the fraction of heads is after the first i tosses. Then to actually plot the fraction of heads vs the number of tosses, enter the following:

```
import matplotlib.pyplot as plt
plt.figure(figsize=(8, 4))
plt.plot(range(1, n+1), fraction_of_heads)
plt.xlabel('Number of flips')
plt.ylabel('Fraction of heads')
```

For example, when we run this we get the following plot:



The fraction of heads initially can be far from $1/2$ but as the number of flips increases, the fraction stabilizes and gets closer to $1/2$, the probability of heads.

Computer note: Many times in this course, it will be helpful to run simulations to test code and plot histograms for different outcomes to get a sense of how likely the outcomes are. Simulations and visualizations can be powerful not only in making sure your code is working correctly but also to present results to people!

1.3 Probability and the Art of Modeling Uncertainty

Since probability effectively corresponds to a fraction, it is a value between 0 and 1. Of course, we can have impossible events that have probability 0, or events that deterministically happen and thus have probability 1. Each time we model uncertainty in the world, there will be some underlying experiment (such as flipping a coin in our running example). An event happens with probability $q \in [0, 1]$ if in a massive number of repeats of the experiment, the event happens roughly a fraction q of the time; more repeats of the experiment make it so that the fraction gets closer to q .

Some times, an underlying experiment cannot possibly be repeated. Take for instance weather forecasting. Whereas we could actually physically flip a coin many times to repeat the same experiment, we cannot physically repeat a real-life experiment for what different realizations of tomorrow's weather will be. We could wait until tomorrow to see the weather, but then we would need a time machine to go back in time by one day to repeat and see what the weather is like tomorrow (and this assumes that there's some inherent randomness in tomorrow's weather)! In such a case, our only hope is to somehow model or simulate tomorrow's weather given measurements up to present time.

Different people could model the same real world problem differently! Throughout the course, a recurring challenge in building computer programs that reason probabilistically is figuring out how to model real-world problems. A good model — even if not actually accurate in describing, for instance, the science behind weather — enables us to make good predictions.

Once a weather forecaster has anchored some way of modeling or simulating weather, then if it claims that there's a 30% chance of rain tomorrow, we could interpret this as saying that using their way of simulating tomorrow's weather, in roughly 30% of simulated results for tomorrow's weather, there is rain.

2 Probability Spaces and Events

2.1 Two Ingredients to Modeling Uncertainty

When we think of an uncertain world, we will always think of there being some underlying experiment of interest. To model this uncertain world, it suffices to keep track of two things:

The set of all possible outcomes for the experiment: this set is called the sample space and is usually denoted by the Greek letter Omega Ω . (For the fair coin flip, there are exactly two possible outcomes: heads, tails. Thus, $\Omega = \{\text{heads}, \text{tails}\}$.)

The probability of each outcome: for each possible outcome, assign a probability that is at least 0 and at most 1. (For the fair coin flip, $\mathbb{P}(\text{heads}) = \frac{1}{2}$ and $\mathbb{P}(\text{tails}) = \frac{1}{2}$.)

Notation: Throughout this course, for any statement \mathcal{S} , “ $\mathbb{P}(\mathcal{S})$ ” denotes the probability of \mathcal{S} happening.

In Python:

```
> model = {'heads': 1/2, 'tails': 1/2}
```

In particular, we see that we can model uncertainty in code using a Python dictionary. The sample space is precisely the keys in the dictionary:

```
> sample_space = set(model.keys())
{'tails', 'heads'}
```

Of course, the dictionary gives us the assignment of probabilities, meaning that for each outcome in the sample space (i.e., for each key in the dictionary), we have an assigned probability:

```
> model['heads']
0.5
> model['tails']
0.5
```

A few important remarks:

- The sample space is always specified to be *collectively exhaustive*, meaning that every possible outcome is in it, and *mutually exclusive*, meaning that once the experiment is run (e.g., flipping the fair coin), exactly one possible outcome in the sample space happens. It’s impossible for multiple outcomes in the sample space to simultaneously happen! It’s also impossible for none of the outcomes to happen!
- Probabilities can be thought of as fractions of times outcomes occur; thus, probabilities are nonnegative and at least 0 and at most 1.
- If we add up the probabilities of all the possible outcomes in the sample space, we get 1. (For the fair coin flip, $\mathbb{P}(\text{heads}) + \mathbb{P}(\text{tails}) = \frac{1}{2} + \frac{1}{2} = 1$.)

Some intuition for this: Consider the coin flipping experiment. What does the fraction of times heads occur and the fraction of times tails occur add up to? Since these are the only two possible outcomes (and again, recall that these outcomes are exclusive in that they can’t simultaneously occur, and exhaustive since they are the only possible outcomes), these two fractions will always sum to 1. For a massive number of repeats of the experiment, these two fractions correspond to $\mathbb{P}(\text{heads})$ and $\mathbb{P}(\text{tails})$; the fractions sum to 1 and so these probabilities also sum to 1.

2.2 Probability Spaces

At this point, we’ve actually already seen the most basic data structure used throughout this course for modeling uncertainty, called a *finite probability space* (in this course, we’ll often also just call this either a *probability space* or a *probability model*):

A *finite probability space* consists of two ingredients:

- a sample space Ω consisting of a *finite* (i.e., not infinite) number of collectively exhaustive and mutually exclusive possible outcomes
- an assignment of probabilities: for each possible outcome $\omega \in \Omega$, we assign a probability $\mathbb{P}(\text{outcome } \omega)$ at least 0 and at most 1, where we require that the probabilities across all the possible outcomes in the sample space add up to 1:

$$\sum_{\omega \in \Omega} \mathbb{P}(\text{outcome } \omega) = 1$$

Notation: As shorthand we occasionally use the tuple “ (Ω, \mathbb{P}) ” to refer to a finite probability space to remind ourselves of the two ingredients needed, sample space Ω and an assignment of probabilities \mathbb{P} . As we already saw, in code these two pieces can be represented together in a single Python dictionary. However, when we want to reason about probability spaces in terms of the mathematics, it’s helpful to have names for the two pieces.

Why finite? Of the two pieces making up a finite probability space (Ω, \mathbb{P}) , the sample space Ω being finite is a fairly natural constraint, corresponding to how we typically work with Python dictionaries where there is only a finite number of keys. As we’ll see, finite probability spaces are already extremely useful in practice. Pedagogically, finite probability spaces also provide a great intro to probability theory as they already carry a wealth of intuition, much of which carries over to a more complete story of general probability spaces!

2.3 Table Representation

A probability space is a data structure in that we can always visualize as a table of nonnegative entries that sum to 1. Let’s see a concrete example of this, first writing the table out on paper and then coding it up.

Example: Suppose we have a model of tomorrow’s weather given as follows: sunny with probability $1/2$, rainy with probability $1/6$, and snowy with probability $1/3$. Here’s the probability space, shown as a table:

	Probability
Outcome sunny	$1/2$
rainy	$1/6$
snowy	$1/3$

Note: This a table of 3 nonnegative entries that sum to 1. The rows correspond to the sample space $\Omega = \{\text{sunny}, \text{rainy}, \text{snowy}\}$.

We will often use this table representation of a probability space to tell you how we’re modeling uncertainty for a particular problem. It provides the simplest of visualizations of a probability space.

Of course, in Python code, the above probability space is given by:

```
prob_space = {'sunny': 1/2, 'rainy': 1/6, 'snowy': 1/3}
```

A different way to code up the same probability space is to separately specify the outcomes (i.e., the sample space) and the probabilities:

```
outcomes = ['sunny', 'rainy', 'snowy']
probabilities = np.array([1/2, 1/6, 1/3])
```

The i -th entry of `outcomes` has probability given by the i -th entry of `probabilities`. Note that `probabilities` is a vector of numbers that we represent as a Numpy array. Numpy has various built-in methods that enable us to easily work with vectors (and more generally arrays) of numbers.

2.4 More on Sample Spaces

In the video, we saw that a sample space encoding the outcomes of 2 coin flips encodes all the information for 1 coin flip as well. Thus, we could use the same sample space to model a single coin flip. However, if we really only cared about a single coin flip, then a sample space encoding 2 coin flips is richer than we actually need it to be!

When we model some uncertain situation, how we specify a sample space is not unique. We saw an example of this already in an earlier exercise where for rolling a single six-sided die, we can choose to name the outcomes differently, saying for instance “roll 1” instead of “1”. We could even add a bunch of extraneous outcomes that all

have probability 0. We could add extraneous information that doesn't matter such as "Alice rolls 1", "Bob rolls 1", etc where we enumerate out all the people who could roll the die in which the outcome is a 1. Sure, depending on the problem we are trying to solve, maybe knowing who rolled the die is important, but if we don't care about who rolled the die, then the information isn't helpful but it's still possible to include this information in the sample space.

Generally speaking it's best to choose a sample space that is as simple as possible for modeling what we care about solving. For example, if we were rolling a six-sided die, and we actually only care about whether the face shows up at least 4 or not, then it's sufficient to just keep track of two outcomes, "at least 4" and "less than 4".

2.5 Probabilities with Events

TODO – add notes from video

2.6 Events as Sets

TODO – add notes from video

2.7 Code for Dealing with Sets in Python

In the video, the set operations can actually be implemented in Python as follows:

```
sample_space = {'HH', 'HT', 'TH', 'TT'}
A = {'HT', 'TT'}
B = {'HH', 'HT', 'TH'}
C = {'HH'}
A_intersect_B = A.intersection(B) # equivalent to "B.intersection(A)" or "A & B"
A_union_C = A.union(C) # equivalent to "C.union(A)" and also "A | C"
B_complement = sample_space.difference(B) # equivalent also to "sample_space - B"
```

2.8 Probabilities with Events and Code

From the videos, we see that an event is a subset of the sample space Ω . If you remember our table representation for a probability space, then an event could be thought of as a subset of the rows, and the probability of the event is just the sum of the probability values in those rows!

The probability of an event $\mathcal{A} \subseteq \Omega$ is the sum of the probabilities of the possible outcomes in \mathcal{A} :

$$\mathbb{P}(\mathcal{A}) \triangleq \sum_{\omega \in \mathcal{A}} \mathbb{P}(\text{outcome } \omega),$$

where " \triangleq " means "defined as".

We can translate the above equation into Python code. In particular, we can compute the probability of an event encoded as a Python set event, where the probability space is encoded as a Python dictionary `prob_space`:

```
def prob_of_event(event, prob_space):
    total = 0
    for outcome in event:
        total += prob_space[outcome]
    return total
```

Here's an example of how to use the above function:

```
prob_space = {'sunny': 1/2, 'rainy': 1/6, 'snowy': 1/3}
rainy_or_snowy_event = {'rainy', 'snowy'}
print(prob_of_event(rainy_or_snowy_event, prob_space))
```


3 Random Variables

3.1 A First Look at Random Variables

Follow along in an IPython prompt.

We continue with our weather example.

```
> prob_space = {'sunny': 1/2, 'rainy': 1/6, 'snowy': 1/3}
```

We can simulate tomorrow's weather using the above model of the world. Let's simulate two different values, one (which we'll call W for "weather") for whether tomorrow will be sunny, rainy, or snowy, and another (which we'll call I for "indicator") that is 1 if it is sunny and 0 otherwise:

```
> random_outcome = comp_prob_inference.sample_from_finite_probability_space(prob_space)
> W = random_outcome
> if random_outcome == 'sunny':
>     I = 1
> else:
>     I = 0
```

Print out the variables W or I to see that they take on specific values. Then re-run the above block of code a few times.

You should see that W and I change and are random (following the probabilities given by the probability space).

This code shows something that's of key importance that we'll see throughout the course. Variables W and I store the values of what are called random variables.

3.2 Random Variables

To mathematically reason about a random variable, we need to somehow keep track of the full range of possibilities for what the random variable's value could be, and how probable different instantiations of the random variable are. The resulting formalism may at first seem a bit odd but as we progress through the course, it will become more apparent how this formalism helps us study real-world problems and address these problems with powerful solutions.

To build up to the formalism, first note, computationally, what happened in the code in the previous part.

1. First, there is an underlying probability space (Ω, \mathbb{P}) , where $\Omega = \{\text{sunny}, \text{rainy}, \text{snowy}\}$, and

$$\mathbb{P}(\text{sunny}) = 1/2, \tag{1}$$

$$\mathbb{P}(\text{rainy}) = 1/6, \tag{2}$$

$$\mathbb{P}(\text{snowy}) = 1/3. \tag{3}$$

2. A random outcome $\omega \in \Omega$ is sampled using the probabilities given by the probability space (Ω, \mathbb{P}) . This step corresponds to an underlying experiment happening.
3. Two random variables are generated:

- W is set to be equal to ω . As an equation:

$$W(\omega) = \omega \text{ for } \omega \in \{\text{sunny}, \text{rainy}, \text{snowy}\}. \tag{4}$$

This step perhaps seems entirely unnecessary, as you might wonder “Why not just call the random outcome W instead of ω ?” Indeed, this step isn’t actually necessary for this particular example, but the formalism for random variables has this step to deal with what happens when we encounter a random variable like I .

- I is set to 1 if $\omega = \text{sunny}$, and 0 otherwise. As an equation:

$$I(\omega) = \begin{cases} 1 & \text{if } \omega = \text{sunny}, \\ 0 & \text{if } \omega \in \{\text{rainy}, \text{snowy}\}. \end{cases} \quad (5)$$

Importantly, multiple possible outcomes (rainy or snowy) get mapped to the same value 0 that I can take on.

We see that random variable W maps the sample space $\Omega = \{\text{sunny}, \text{rainy}, \text{snowy}\}$ to the same set $\{\text{sunny}, \text{rainy}, \text{snowy}\}$. Meanwhile, random variable I maps the sample space $\Omega = \{\text{sunny}, \text{rainy}, \text{snowy}\}$ to the set $\{0, 1\}$.

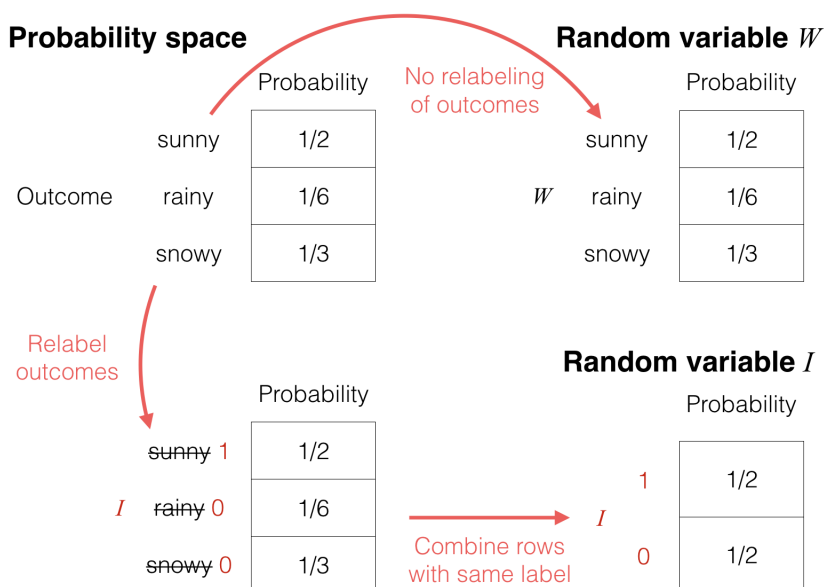
In general:

Definition of a “finite random variable” (in this course, we will just call this a “random variable”): Given a finite probability space (Ω, \mathbb{P}) , a finite random variable X is a mapping from the sample space Ω to a set of values X that random variable X can take on. (We will often call X the “alphabet” of random variable X .)

For example, random variable W takes on values in the alphabet sunny,rainy,snowy, and random variable I takes on values in the alphabet $\{0, 1\}$.

Quick summary: There’s an underlying experiment corresponding to probability space (Ω, \mathbb{P}) . Once the experiment is run, let $\omega \in \Omega$ denote the outcome of the experiment. Then the random variable takes on the specific value of $X(\omega) \in \mathcal{X}$.

Explanation using a picture: Continuing with the weather example, we can pictorially see what’s going on by looking at the probability tables for: the original probability space, the random variable W , and the random variable I :



These tables make it clear that a “random variable” really is just reassigning/relabeling what the values are for the possible outcomes in the underlying probability space (given by the top left table):

- In the top right table, random variable W does not do any sort of relabeling so its probability table looks the same as that of the underlying probability space.
- In the bottom left table, the random variable I relabels/reassigns “sunny” to 1, and both “rainy” and “snowy” to 0. Intuitively, since two of the rows now have the same label 0, it makes sense to just combine these two rows, adding their probabilities $\frac{1}{6} + \frac{1}{3} = \frac{1}{2}$. This results in the bottom right table.

Technical note: Even though the formal definition of a finite random variable doesn’t actually make use of the probability assignment \mathbb{P} , the probability assignment will become essential as soon as we talk about how probability works with random variables.

3.3 Random Variables Notation and Terminology

In this course, we denote random variables with capital/uppercase letters, such as X , W , I , etc. We use the phrases “probability table”, “probability mass function” (abbreviated as PMF), and “probability distribution” (often simply called a distribution) to mean the same thing, and in particular we denote the probability table for X to be p_X or $p_X(\cdot)$.

We write $p_X(x)$ to denote the entry of the probability table that has label $x \in \mathcal{X}$ where \mathcal{X} is the set of values that random variable \mathcal{X} takes on. Note that we use lowercase letters like x to denote variables storing nonrandom values. We can also look up values in a probability table using specific outcomes, e.g., from earlier, we have $p_W(\text{rainy}) = 1/6$ and $p_I(1) = 1/2$.

Note that we use the same notation as in math where a function f might also be written as $f(\cdot)$ to explicitly indicate that it is the function of one variable. Both f and $f(\cdot)$ refer to a function whereas $f(x)$ refers to the value of the function f evaluated at the point x .

As an example of how to use all this notation, recall that a probability table consists of nonnegative entries that add up to 1. In fact, each of the entries is at most 1 (otherwise the numbers would add to more than 1). For a random variable X taking on values in \mathcal{X} , we can write out these constraints as:

$$0 \leq p_X(x) \leq 1 \quad \text{for all } x \in \mathcal{X}, \quad \sum_{x \in \mathcal{X}} p_X(x) = 1.$$

Often in the course, if we are making statements about all possible outcomes of X , we will omit writing out the alphabet \mathcal{X} explicitly. For example, instead of the above, we might write the following equivalent statement:

$$0 \leq p_X(x) \leq 1 \quad \text{for all } x, \quad \sum_x p_X(x) = 1.$$

4 Jointly Distributed Random Variables

4.1 Relating Two Random Variables

At the most basic level, inference refers to using an observation to reason about some unknown quantity. In this course, the observation and the unknown quantity are represented by random variables. The main modeling question is: How do these random variables relate?

Let’s build on our earlier weather example, where now another outcome of interest appears, the temperature, which we quantize into two possible values “hot” and “cold”. Let’s suppose that we have the following probability space:

Outcome	Probability	
	sunny, hot	3/10
	sunny, cold	1/5
	rainy, hot	1/30
	rainy, cold	2/15
	snowy, hot	0
	snowy, cold	1/3

You can check that the nonnegative entries do add to 1. If we let random variable W be the weather (sunny, rainy, snowy) and random variable T be the temperature (hot, cold), then notice that we could rearrange the table in the following fashion:

Outcome	Probability	
	$W = \text{sunny}, T = \text{hot}$	3/10
	$W = \text{sunny}, T = \text{cold}$	1/5
	$W = \text{rainy}, T = \text{hot}$	1/30
	$W = \text{rainy}, T = \text{cold}$	2/15
	$W = \text{snowy}, T = \text{hot}$	0
	$W = \text{snowy}, T = \text{cold}$	1/3

Rearrange
table entries



		T	
		hot	cold
W	sunny	3/10	1/5
	rainy	1/30	2/15
	snowy	0	1/3

When we talk about two separate random variables, we could view them either as a single “super” random variable that happens to consist of a pair of values (the first table; notice the label for each outcome corresponds to a pair of values), or we can view the two separate variables along their own different axes (the second table).

The first table tells us what the underlying probability space is, which includes what the sample space is (just read off the outcome names) and what the probability is for each of the possible outcomes for the underlying experiment at hand.

The second table is called a joint probability table $p_{W,T}$ for random variables W and T , and we say that random variables W and T are jointly distributed with the above distribution. Since this table is a rearrangement of the earlier table, it also consists of nonnegative entries that add to 1.

The joint probability table gives probabilities in which W and T co-occur with specific values. For example, in the above, the event that “ $W = \text{sunny}$ ” and the event that “ $T = \text{hot}$ ” co-occur with probability 3/10. Notationally, we write

$$p_{W,T}(\text{sunny}, \text{hot}) = \mathbb{P}(W = \text{sunny}, T = \text{hot}) = \frac{3}{10}.$$

Conceptual note: Given the joint probability table, we can easily go backwards and write out the first table above, which is the underlying probability space.

Preview of inference: Inference is all about answering questions like “if we observe that the weather is rainy, what is the probability that the temperature is cold?” Let’s take a look at how one might answer this question.

First, if we observe that it is rainy, then we know that “sunny” and “snowy” didn’t happen so those rows aren’t relevant anymore. So the space of possible realizations of the world has shrunk to two options now: ($W = \text{rainy}, T = \text{hot}$) or ($W = \text{rainy}, T = \text{cold}$). But what about the probabilities of these two realizations? It’s not just $1/30$ and $2/15$ since these don’t sum to 1 — by observing things, adjustments can be made to the probabilities of different realizations but they should still form a valid probability space.

Why not just scale both $1/30$ and $2/15$ by the same constant so that they sum to 1? This can be done by dividing $1/30$ and $2/15$ by their sum:

$$\text{hot: } \frac{\frac{1}{30}}{\frac{1}{30} + \frac{2}{15}} = \frac{1}{5}, \quad \text{cold: } \frac{\frac{2}{15}}{\frac{1}{30} + \frac{2}{15}} = \frac{4}{5}.$$

Now they sum to 1. It turns out that, given that we’ve observed the weather to be rainy, these are the correct probabilities for the two options “hot” and “cold”. Let’s formalize the steps. We work backwards, first explaining what the the denominator “ $\frac{1}{30} + \frac{2}{15} = \frac{1}{6}$ ” above comes from.

4.2 Representing a Joint Probability Table in Code

There are various ways to represent a joint probability table in code. Here are a few.

Note that we have updated `comp_prob_inference.py`! Please re-download it!

Approach 0: Don’t actually represent the joint probability table. This doesn’t store the 2D table at all but is a first attempt at coding up something that has all the information in the joint probability table. Specifically, we can just code up the entries like how we coded up a probability space:

```
> prob_table = {('sunny', 'hot'): 3/10,
>   ('sunny', 'cold'): 1/5,
>   ('rainy', 'hot'): 1/30,
>   ('rainy', 'cold'): 2/15,
>   ('snowy', 'hot'): 0,
>   ('snowy', 'cold'): 1/3}
```

Thus, if we want the probability of $W = \text{rainy}$ and $T = \text{cold}$, we write:

```
> prob_table[('rainy', 'cold')]
0.13333333333333333
```

Some times, this representation is sufficient. Given a specific weather and temperature stored as strings in `w` and `t` respectively, `prob_table[(w, t)]` gives you the joint probability table evaluated at $W = w$ and $T = t$.

Approach 1: Use dictionaries within a dictionary. This works as follows:

```
> prob_W_T_dict = {}
> for w in {'sunny', 'rainy', 'snowy'}:
>     prob_W_T_dict[w] = {}
>
> prob_W_T_dict['sunny']['hot'] = 3/10
> prob_W_T_dict['sunny']['cold'] = 1/5
> prob_W_T_dict['rainy']['hot'] = 1/30
> prob_W_T_dict['rainy']['cold'] = 2/15
> prob_W_T_dict['snowy']['hot'] = 0
> prob_W_T_dict['snowy']['cold'] = 1/3
>
> comp_prob_inference.print_joint_prob_table_dict(prob_W_T_dict)
      cold      hot
rainy 0.133333 0.033333
```

```
snowy 0.333333 0.000000
sunny 0.200000 0.300000
```

Note that because dictionary keys aren't ordered, the row ordering and column ordering need not match the tables we have been showing in the course notes earlier. This is not a problem.

If we want the probability of $W = \text{rainy}$ and $T = \text{cold}$, we write:

```
> prob_W_T_dict['rainy']['cold']
0.13333333333333333
```

The probability for $W = w$ and $T = t$ is stored in `prob_W_T_dict[w][t]`.

Approach 2: Use a 2D array. Another approach is to directly store the joint probability table as a 2D array, separately keeping track of what the rows and columns are. We use a NumPy array (but really you could use Python lists within a Python list, much like how the previous approach used dictionaries within a dictionary; the indexing syntax changes only slightly):

```
> import numpy as np
> prob_W_T_rows = ['sunny', 'rainy', 'snowy']
> prob_W_T_cols = ['hot', 'cold']
> prob_W_T_array = np.array([[3/10, 1/5], [1/30, 2/15], [0, 1/3]])
> comp_prob_inference.print_joint_prob_table_array(prob_W_T_array, prob_W_T_rows, prob_W_T_cols)
      hot      cold
sunny 0.300000 0.200000
rainy 0.033333 0.133333
snowy 0.000000 0.333333
```

Note that the ordering of rows is specified, as is the ordering of the columns, unlike in the dictionaries within a dictionary representation.

Retrieving a specific table entry requires a little bit more code since we need to figure out what the row and column numbers are corresponding to a specific pair of row and column labels. For example, if we want the probability of $W = \text{rainy}$ and $T = \text{cold}$, we write:

```
> prob_W_T_array[prob_W_T_rows.index('rainy'), prob_W_T_cols.index('cold')]
0.13333333333333333
```

Note that `prob_W_T_rows.index('rainy')` finds the row number (starting from 0) corresponding to “rainy”.

Using `.index` does a search through the whole list of row/column labels, which for large lists can be slow. Let's fix this!

A cleaner and faster way is to create separate dictionaries mapping the row and column labels to row and column indices in the 2D array. In other words, instead of writing `prob_W_T_rows.index('rainy')` to find the row number for 'rainy', we want to just be able to write something like `prob_W_T_row_mapping['rainy']`, which returns the row number. We can define Python variable `prob_W_T_row_mapping` as follows:

```
> prob_W_T_row_mapping = {}
> for index, label in enumerate(prob_W_T_rows):
>     prob_W_T_row_mapping[label] = index
```

Note that `enumerate(prob_W_T_rows)` produces an iterator that consists of pairs $(0, \text{'sunny'})$, $(1, \text{'rainy'})$, $(2, \text{'snowy'})$. You can see this by entering:

```
> print(list(enumerate(prob_W_T_rows)))
[(0, 'sunny'), (1, 'rainy'), (2, 'snowy')]
```

Note that each pair consists of the row number followed by the label.

In fact, the three lines we used to define `prob_W_T_row_mapping` can be written in one line with a Python dictionary comprehension:

```
> prob_W_T_row_mapping = {label: index for index, label in enumerate(prob_W_T_rows)}
```

We can do the same thing to define a mapping of column labels to column numbers:

```
> prob_W_T_col_mapping = {label: index for index, label in enumerate(prob_W_T_cols)}
```

In summary, we can represent the joint probability table as follows:

```
> prob_W_T_rows = ['sunny', 'rainy', 'snowy']
> prob_W_T_cols = ['hot', 'cold']
> prob_W_T_row_mapping = {label: index for index, label in enumerate(prob_W_T_rows)}
> prob_W_T_col_mapping = {label: index for index, label in enumerate(prob_W_T_cols)}
> prob_W_T_array = np.array([[3/10, 1/5], [1/30, 2/15], [0, 1/3]])
```

Now the probability for $W = w$ and $T = t$ is given by:

```
> prob_W_T_array[prob_W_T_row_mapping[w], prob_W_T_col_mapping[t]]
```

Some remarks: The 2D array representation, as we'll see soon, is very easy to work with when it comes to basic operations like summing rows, and retrieving a specific row or column. The main disadvantage of this representation is that you need to store the whole array, and if the alphabet sizes of the random variables are very large, then storing the array will take a lot of space!

The dictionaries within a dictionary representation allows for easily retrieving rows but not columns (try it: write a Python function that picks out a specific row and another function that picks out a specific column; you should see that retrieving a row is easier because it corresponds to looking at the value stored for a single key of the outer dictionary). This also means that summing a column's probabilities is more cumbersome than summing a row's probabilities. However, a huge advantage of this way of representing a joint probability table is that in many problems, we have a massive joint probability table that is mostly filled with 0's. Thus, the 2D array representation would require storing a very, very large table with many 0's, whereas the dictionaries within a dictionary representation is able to only store the nonzero table entries. We'll see more about this issue when we look at robot localization in the second section of the course on inference in graphical models.

4.3 Marginalization

Given a joint probability table, often we'll want to know what the probability table is for just one of the random variables. We can do this by just summing or "marginalizing" out the other random variables. For example, to get the probability table for random variable W , we do the following:

		T					
		hot	cold			Prob.	
W	sunny	3/10	1/5	1/2	→	sunny	1/2
	rainy	1/30	2/15	1/6		rainy	1/6
	snowy	0	1/3	1/3		snowy	1/3
		Add up each row		Numbers in right margin form table p_W			

We take the joint probability table (left-hand side) and compute out the row sums (which we've written in the

margin).

The right-hand side table is the probability table p_W for random variable W ; we call this resulting probability distribution the marginal distribution of W (put another way, it is the distribution obtained by marginalizing out the random variables that aren't W).

In terms of notation, the above marginalization procedure whereby we used the joint distribution of W and T to produce the marginal distribution of W is written:

$$p_W(w) = \sum_{t \in \mathcal{T}} p_{W,T}(w, t),$$

where \mathcal{T} is the set of values that random variable T can take on. In fact, throughout this course, we will often omit explicitly writing out the alphabet of values that a random variable takes on, e.g., writing instead



$$p_W(w) = \sum_t p_{W,T}(w, t).$$

It's clear from context that we're summing over all possible values for t , which is going to be the values that random variable T can possibly take on.

As a specific example,

$$p_W(\text{rainy}) = \sum_t p_{W,T}(\text{rainy}, t) = \underbrace{p_{W,T}(\text{rainy}, \text{hot})}_{1/30} + \underbrace{p_{W,T}(\text{rainy}, \text{cold})}_{2/15} = \frac{1}{6}.$$

We could similarly marginalize out random variable W to get the marginal distribution p_T for random variable T :

		T		
		hot	cold	
W	sunny	3/10	1/5	 Add up each column
	rainy	1/30	2/15	
	snowy	0	1/3	
		1/3	2/3	
		 Numbers in bottom margin form table p_T		
		T		
		hot	cold	
Prob.		1/3	2/3	

(Note that whether we write a probability table for a single variable horizontally or vertically doesn't actually matter.)

As a formula, we would write:

$$p_T(t) = \sum_w p_{W,T}(w, t).$$

For example,

$$p_T(\text{hot}) = \sum_w p_{W,T}(w, \text{hot}) = \underbrace{p_{W,T}(\text{sunny}, \text{hot})}_{3/10} + \underbrace{p_{W,T}(\text{rainy}, \text{hot})}_{1/30} + \underbrace{p_{W,T}(\text{snowy}, \text{hot})}_0 = \frac{1}{3}.$$

In general:

Marginalization: Consider two random variables X and Y (that take on values in the sets \mathcal{X} and \mathcal{Y} respectively) with joint probability table $p_{X,Y}$. For any $x \in \mathcal{X}$, the marginal probability that $X = x$ is given by

$$p_X(x) = \sum_y p_{X,Y}(x, y).$$

4.4 Marginalization for Many Random Variables

What happens when we have more than two random variables? Let's build on our earlier example and suppose that in addition to weather W and temperature T , we also had a random variable H for humidity that takes on values in the alphabet dry, humid. Then having a third random variable, we can draw out a 3D joint probability table for random variables W , T , and H . As an example, we could have the following:

		T		
		hot	cold	
W	sunny	1/5	1/10	dry humid H
	rainy	1/10	1/10	
	snowy	1/30	2/15	
		0	2/9	1/9

Here, each of the cubes/boxes stores a probability. Not visible are two of the cubes in the back left column, which for this particular example both have probability values of 0.

Then to marginalize out the humidity H , we would add values as follows:

		T		
		hot	cold	
W	sunny	1/5	1/10	dry humid H
	rainy	1/10	1/10	
	snowy	1/30	2/15	
		0	2/9	1/9

Add along this direction to marginalize out humidity

		T	
		hot	cold
W	sunny	3/10	1/5
	rainy	1/30	2/15
	snowy	0	1/3

The resulting numbers form the table $p_{W,T}$

The result is the joint probability table for weather W and temperature T , shown still in 3D cubes with each cube storing a single probability.

As an equation:

$$p_{W,T}(w,t) = \sum_h p_{W,T,H}(w,t,h).$$

In general, for three random variables X , Y , and Z with joint probability table $p_{X,Y,Z}$, we have

$$p_{X,Y}(x,y) = \sum_z p_{X,Y,Z}(x,y,z), \quad (6)$$

$$p_{X,Z}(x,z) = \sum_y p_{X,Y,Z}(x,y,z), \quad (7)$$

$$p_{Y,Z}(y,z) = \sum_x p_{X,Y,Z}(x,y,z). \quad (8)$$

Note that we can marginalize out different random variables in succession. For example, given joint probability table $p_{X,Y,Z}$, if we wanted the probability table p_X , we can get it by marginalizing out the two random variables Y and Z :

$$p_X(x) = \sum_y p_{X,Y}(x,y) = \sum_y \left(\sum_z p_{X,Y,Z}(x,y,z) \right).$$

Even with more than three random variables, the idea is the same. For example, with four random variables W , X , Y , and Z with joint probability table $p_{W,X,Y,Z}$, if we want the joint probability table for X and Y , we would do the following:

$$p_{X,Y}(x,y) = \sum_w \left(\sum_z p_{W,X,Y,Z}(w,x,y,z) \right).$$

4.5 Conditioning for Random Variables

When we observe that a random variable takes on a specific value (such as $W = \text{rainy}$ from earlier for which we say that we condition on random variable W taking on the value “rainy”), this observation can affect what we think are likely or unlikely values for another random variable.

When we condition on $W = \text{rainy}$, we do a two-step procedure; first, we only keep the row for W corresponding to the observed value:

	T			T	
	hot	cold		hot	cold
W sunny	3/10	1/5	Keep row for $W = \text{rainy}$ →		
W rainy	1/30	2/15		W rainy	1/30 2/15
W snowy	0	1/3			
				Not valid prob. distribution (since sum $\neq 1$)	

Second, we “normalize” the table so that its entries add up to 1, which corresponds to dividing it by the sum of the entries, which is equal to $p_W(\text{rainy})$ in this case:

	T			T	
	hot	cold		hot	cold
W rainy	1/30	2/15	Rescale entries so they add to 1 →	W rainy	1/5 4/5

Notation: The resulting probability table $p_{T|W}(\cdot | \text{rainy})$ is associated with the random variable denoted $(T | W = \text{rainy})$; we use “|” to denote that we’re conditioning on things to the right of “|” happening (these are things that we have observed or that we are given as having happened). We read “ $T | W = \text{rainy}$ ” as either “ T given W is rainy” or “ T conditioned on W being rainy”. To refer to specific entries of the table, we write, for instance,

$$p_{T|W}(\text{cold} | \text{rainy}) = \mathbb{P}(T = \text{cold} | W = \text{rainy}) = \frac{4}{5}.$$

In general:

Conditioning: Consider two random variables X and Y (that take on values in the sets \mathcal{X} and \mathcal{Y} respectively) with joint probability table $p_{X,Y}$ (from which by marginalization we can readily compute the marginal probability table p_Y). For any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ such that $p_Y(y) > 0$, the conditional probability of event $X = x$ given event $Y = y$ has happened is

$$p_{X|Y}(x | y) \triangleq \frac{p_{X,Y}(x,y)}{p_Y(y)}.$$

For example,

$$p_{T|W}(\text{cold} | \text{rainy}) = \frac{p_{W,T}(\text{rainy}, \text{cold})}{p_W(\text{rainy})} = \frac{\frac{2}{15}}{\frac{1}{6}} = \frac{4}{5}.$$

Computational interpretation: To compute $p_{X|Y}(x | y)$, take the entry $p_{X,Y}(x,y)$ in the joint probability table corresponding to $X = x$ and $Y = y$, and then divide the entry by $p_Y(y)$, which is an entry in the marginal probability table p_Y for random variable Y .

4.6 Moving Toward a More General Story for Conditioning

Jointly distributed random variables play a central role in this course. Remember that we will model observations as random variables and the quantities we want to infer also as random variables. When these random variables are jointly distributed so that we have a probabilistic way to describe how they relate (through their joint probability table), then we can systematically and quantitatively produce inferences.

We just saw how to condition on a random variable taking on a specific value. What about if we wanted to condition on a random variable taking on any one of many values rather just one specific value? To answer this question, we look at a more general story of conditioning which is in terms of events.

5 Conditioning on Events

5.1 Conditioning on Events Intro

TODO – add notes from video

5.2 The Product Rule for Events

TODO – add notes from video

5.3 Bayes' Theorem for Events

Important note about dividing by probabilities: We will often divide by probabilities. In videos, we might not always say this, but this is required: we cannot divide by 0. To ensure this, we will not condition on events that have probability 0.

Given two events \mathcal{A} and \mathcal{B} (both of which have positive probability), Bayes' theorem, also called Bayes' rule or Bayes' law, gives a way to compute $\mathbb{P}(\mathcal{A}|\mathcal{B})$ in terms of $\mathbb{P}(\mathcal{B}|\mathcal{A})$. This result turns out to be extremely useful for inference because often times we want to compute one of these, and the other is known or otherwise straightforward to compute.

Bayes' theorem is given by

$$\mathbb{P}(\mathcal{A}|\mathcal{B}) = \frac{\mathbb{P}(\mathcal{B}|\mathcal{A})\mathbb{P}(\mathcal{A})}{\mathbb{P}(\mathcal{B})}.$$

The proof of why this is the case is a one liner:

$$\mathbb{P}(\mathcal{A}|\mathcal{B}) \stackrel{(a)}{=} \frac{\mathbb{P}(\mathcal{A} \cap \mathcal{B})}{\mathbb{P}(\mathcal{B})} \stackrel{(b)}{=} \frac{\mathbb{P}(\mathcal{B}|\mathcal{A})\mathbb{P}(\mathcal{A})}{\mathbb{P}(\mathcal{B})},$$

where step (a) is by the definition of conditional probability for events, and step (b) is due to the product rule for events (which follows from rearranging the definition of conditional probability for $\mathbb{P}(\mathcal{B}|\mathcal{A})$).

5.4 Practice Problem: Bayes' Theorem and Total Probability

Your problem set is due in 15 minutes! It's in one of your drawers, but they are messy, and you're not sure which one it's in.

The probability that the problem set is in drawer k is d_k . If drawer k has the problem set and you search there, you have probability p_k of finding it. There are a total of m drawers.

Suppose you search drawer i and do not find the problem set.

- (a) Find the probability that the paper is in drawer j , where $j \neq i$.
- (b) Find the probability that the paper is in drawer i .

Solution: Let A_k be the event that the problem set is in drawer k , and B_k be the event that you find the problem set in drawer k .

(a) We'll express the desired probability as $\mathbb{P}(A_j|B_i^c)$. Since this quantity is difficult to reason about directly, we'll use Bayes' rule:

$$\mathbb{P}(A_j|B_i^c) = \frac{\mathbb{P}(B_i^c|A_j)\mathbb{P}(A_j)}{\mathbb{P}(B_i^c)}$$

The first probability, $\mathbb{P}(B_i^c|A_j)$, expresses the probability of not finding the problem set in drawer i given that it's in a different drawer j . Since it's impossible to find the paper in a drawer it isn't in, this is just 1.

The second quantity, $\mathbb{P}(A_j)$, is given to us in the problem statement as d_j .

The third probability, $\mathbb{P}(B_i^c) = 1 - \mathbb{P}(B_i)$, is difficult to reason about directly. But, if we knew whether or not the paper was in the drawer, it would become easier. So, we'll use total probability:

$$\mathbb{P}(B_i) = \mathbb{P}(B_i|A_i)\mathbb{P}(A_i) + \mathbb{P}(B_i|A_i^c)\mathbb{P}(A_i^c) \quad (9)$$

$$= p_i d_i + 0(1 - d_i) \quad (10)$$

Putting these terms together, we find that

$$\mathbb{P}(A_j|B_i^c) = \frac{d_j}{1 - p_i d_i}$$

Alternate method to compute the denominator $\mathbb{P}(B_i^c)$: We could use the law of total probability to decompose $\mathbb{P}(B_i^c)$ depending on which drawer the homework is actually in. We have

$$\mathbb{P}(B_i^c) = \sum_{k=1}^m \underbrace{\mathbb{P}(A_k)}_{d_k} \underbrace{\mathbb{P}(B_i^c|A_k)}_{\substack{1 \text{ if } k \neq i, \\ (1-p_i) \text{ if } k=i}} \quad (11)$$

$$= \sum_{\substack{k=1, \\ k \neq i}}^m d_k + (1 - p_i)d_i \quad (12)$$

$$= \sum_{k=1}^m d_k - p_i d_i \quad (13)$$

$$= 1 - p_i d_i. \quad (14)$$

(b) Similarly, we'll use Bayes' rule:

$$\mathbb{P}(A_i|B_i^c) = \frac{\mathbb{P}(B_i^c|A_i)\mathbb{P}(A_i)}{\mathbb{P}(B_i^c)} = \frac{(1-p_i)d_i}{1-p_i d_i}$$

5.5 Take-Away Lessons:

- Defining the sample-space is not always going to help solve the problem. (It's difficult to precisely define the sample space for this particular problem)
- When in doubt of being able to precisely define the sample space, try to define events intelligently, i.e., in a way that you use what you're given in the problem.
- The probability law of a probability model is a function on events, or subsets of the sample space, i.e., one can work with the probability law without knowing precisely what the sample-space (as a set) is.

6 Inference with Bayes' Theorem for Random Variables

6.1 The Product Rule for Random Variables (Also Called the Chain Rule)

In many real world problems, we aren't given what the joint distribution of two random variables is although we might be given other information from which we can compute the joint distribution. Often times, we can compute out the joint distribution using what's called the product rule (often also called the chain rule). This is precisely the random variable version of the product rule for events.

As we saw from before, we were able to derive Bayes' theorem for events using the product rule for events: $\mathbb{P}(\mathcal{A} \cap \mathcal{B}) = \mathbb{P}(\mathcal{A})\mathbb{P}(\mathcal{B} | \mathcal{A})$. The random variable version of the product rule is derived just like the event version of the product rule, by rearranging the equation for the definition of conditional probability. For two random variables X and Y (that take on values in sets \mathcal{X} and \mathcal{Y} respectively), the product rule for random variables says that

$$p_{X,Y}(x,y) = p_Y(y)p_{X|Y}(x|y) \quad \text{for all } x \in \mathcal{X}, y \in \mathcal{Y} \text{ such that } p_Y(y) > 0.$$

Interpretation: If we have the probability table for Y , and separately the probability table for X conditioned on Y , then we can come up with the joint probability table (i.e., the joint distribution) of X and Y .

What happens when $p_Y(y) = 0$? Even though $p_{X|Y}(x|y)$ isn't defined in this case, one can readily show that $p_{X,Y}(x,y) = 0$ when $p_Y(y) = 0$.

To see this, think about what is happening computationally: Remember how $p_Y(y)$ is computed from joint probability table $p_{X,Y}$? In particular, we have $p_Y(y) = \sum_x p_{X,Y}(x,y)$, so $p_Y(y)$ is the sum of either a row or a column in the joint probability table (whether it's a row or column just depends on how you write out the table and which random variable is along which axis along rows or columns). So if $p_Y(y) = 0$, it must mean that the individual elements being summed are 0 (since the numbers we're summing up are nonnegative).

We can formalize this intuition with a proof:

Claim: Suppose that random variables X and Y have joint probability table $p_{X,Y}$ and take on values in sets \mathcal{X} and \mathcal{Y} respectively. Suppose that for a specific choice of $y \in \mathcal{Y}$, we have $p_Y(y) = 0$. Then

$$p_{X,Y}(x,y) = 0 \quad \text{for all } x \in \mathcal{X}.$$

Proof: Let $y \in \mathcal{Y}$ satisfy $p_Y(y) = 0$. Recall that we relate marginal distribution p_Y to joint distribution $p_{X,Y}$ via marginalization:

$$0 = p_Y(y) = \sum_{x \in \mathcal{X}} p_{X,Y}(x,y).$$

Next, we use a crucial mathematical observation: If a sum of nonnegative numbers (such as probabilities) equals 0, then each of the numbers being summed up must also be 0 (otherwise, the sum would be positive!). Hence, it must be that each number being added up in the right-hand side sum is 0, i.e.,

$$p_{X,Y}(x,y) = 0 \quad \text{for all } x \in \mathcal{X}.$$

This completes the proof. \square

Thus, in general:

$$p_{X,Y}(x,y) = \begin{cases} p_Y(y)p_{X|Y}(x|y) & \text{if } p_Y(y) > 0, \\ 0 & \text{if } p_Y(y) = 0. \end{cases}$$

Important convention for this course: For notational convenience, throughout this course, we will often just write $p_{X,Y}(x,y) = p_Y(y)p_{X|Y}(x|y)$ with the understanding that if $p_Y(y) = 0$, even though $p_{X|Y}(x|y)$ is not actually defined, $p_{X,Y}(x,y)$ just evaluates to 0 anyways.

The product rule is symmetric: We can use the definition of conditional probability with X and Y swapped, and rearranging factors, we get:

$$p_{X,Y}(x,y) = p_X(x)p_{Y|X}(y|x) \quad \text{for all } x \in \mathcal{X}, y \in \mathcal{Y} \text{ such that } p_X(x) > 0,$$

and so similarly we could show that

$$p_{X,Y}(x,y) = \begin{cases} p_X(x)p_{Y|X}(y|x) & \text{if } p_X(x) > 0, \\ 0 & \text{if } p_X(x) = 0. \end{cases}$$

Again for notational convenience, we'll typically just write $p_{X,Y}(x,y) = p_X(x)p_{Y|X}(y|x)$ with the understanding that the expression is 0 when $p_Y(y) = 0$.

Interpretation: If we're given the probability table for X and, separately, the probability table for Y conditioned on X , then we can come up with the joint probability table for X and Y .

Importantly, for any two jointly distributed random variables X and Y , the product rule is always true, without making any further assumptions! Also, as a recurring theme that we'll see later on as well, we are decomposing the joint distribution into the product of factors (in this case, the product of two factors).

Many random variables: If we have many random variables, say, X_1, X_2 , up to X_N where N is not a random variable but is a fixed constant, then we have

$$\begin{aligned} & p_{X_1, X_2, \dots, X_N}(x_1, x_2, \dots, x_N) \\ &= p_{X_1}(x_1)p_{X_2|X_1}(x_2|x_1)p_{X_3|X_1, X_2}(x_3|x_1, x_2) \\ & \quad \cdots p_{X_N|X_1, X_2, \dots, X_{N-1}}(x_N|x_1, x_2, \dots, x_{N-1}). \end{aligned}$$

Again, we write this to mean that this holds for every possible choice of x_1, x_2, \dots, x_N for which we never condition on a zero probability event. Note that the above factorization always holds without additional assumptions on the distribution of X_1, X_2, \dots, X_N .

Note that the product rule could be applied in arbitrary orderings. In the above factorization, you could think of it as introducing random variable X_1 first, and then X_2 , and then X_3 , etc. Each time we introduce another random variable, we have to condition on all the random variables that have already been introduced.

Since there are N random variables, there are $N!$ different orderings in which we can write out the product rule. For example, we can think of introducing the last random variable X_N first and then going backwards until we introduce X_1 at the end. This yields the, also correct, factorization

$$\begin{aligned} p_{X_1, X_2, \dots, X_N}(x_1, x_2, \dots, x_N) \\ = p_{X_N}(x_N) p_{X_{N-1}|X_N}(x_{N-1} | x_N) p_{X_{N-2}|X_{N-1}, X_N}(x_{N-2} | x_{N-1}, x_N) \\ \cdots p_{X_1|X_2, X_3, \dots, X_N}(x_1 | x_2, \dots, x_N). \end{aligned}$$

6.2 Bayes' Rule for Random Variables (Also Called Bayes' Theorem for Random Variables)

In inference, what we want to reason about is some unknown random variable X , where we get to observe some other random variable Y , and we have some model for how X and Y relate. Specifically, suppose that we have some "prior" distribution p_X for X ; this prior distribution encodes what we believe to be likely or unlikely values that X takes on, before we actually have any observations. We also suppose we have a "likelihood" distribution $p_{Y|X}$.

After observing that Y takes on a specific value y , our "belief" of what X given $Y = y$ is now given by what's called the "posterior" distribution $p_{X|Y}(\cdot | y)$. Put another way, we keep track of a probability distribution that tells us how plausible we think different values X can take on are. When we observe data Y that can help us reason about X , we proceed to either upweight or downweight how plausible we think different values X can take on are, making sure that we end up with a probability distribution giving us our updated belief of what X can be.

Thus, once we have observed $Y = y$, our belief of what X is changes from the prior p_X to the posterior $p_{X|Y}(\cdot | y)$.

Bayes' theorem (also called Bayes' rule or Bayes' law) for random variables explicitly tells us how to compute the posterior distribution $p_{X|Y}(\cdot | y)$, i.e., how to weight each possible value that random variable X can take on, once we've observed $Y = y$. Bayes' theorem is the main workhorse of numerous inference algorithms and will show up many times throughout the course.

Bayes' theorem: Suppose that y is a value that random variable Y can take on, and $p_Y(y) > 0$. Then

$$p_{X|Y}(x | y) = \frac{p_X(x) p_{Y|X}(y|x)}{\sum_{x'} p_X(x') p_{Y|X}(y|x')}$$

for all values x that random variable X can take on.

Important: Remember that $p_{X|Y}(\cdot | y)$ could be undefined but this isn't an issue since this happens precisely when $p_X(x) = 0$, and we know that $p_{X,Y}(x, y) = 0$ (for every y) whenever $p_X(x) = 0$.

Proof: We have

$$p_{X|Y}(x | y) \stackrel{(a)}{=} \frac{p_{X,Y}(x, y)}{p_Y(y)} \stackrel{(b)}{=} \frac{p_X(x) p_{Y|X}(y|x)}{p_Y(y)} \stackrel{(c)}{=} \frac{p_X(x) p_{Y|X}(y|x)}{\sum_{x'} p_X(x') p_{Y|X}(y|x')} \stackrel{(d)}{=} \frac{p_X(x) p_{Y|X}(y|x)}{\sum_{x'} p_X(x') p_{Y|X}(y|x')},$$

where step (a) uses the definition of conditional probability (this step requires $p_Y(y) > 0$), step (b) uses the product rule (recall that for notational convenience we're not separately writing out the case when $p_X(x) = 0$), step (c) uses the formula for marginalization, and step (d) uses the product rule (again, for notational convenience, we're not separately writing out the case when $p_X(x') = 0$). \square

6.3 Bayes' Theorem for Random Variables: A Computational View

Computationally, Bayes' theorem can be thought of as a two-step procedure. Once we have observed $Y = y$:

For each value x that random variable X can take on, initially we believed that $X = x$ with a score of $p_X(x)$, which could be thought of as how plausible we thought ahead of time that $X = x$. However now that we have observed $Y = y$, we weight the score $p_X(x)$ by a factor $p_{Y|X}(y|x)$, so

new belief for how plausible $X = x$ is: $\alpha(x|y) \triangleq p_X(x)p_{Y|X}(y|x)$,

where we have defined a new table $\alpha(\cdot|y)$ which is not a probability table, since when we put in the weights, the new beliefs are no longer guaranteed to sum to 1 (i.e., $\sum_x \alpha(x|y)$ might not equal 1)! $\alpha(\cdot|y)$ is an unnormalized posterior distribution!

Also, if $p_X(x)$ is already 0, then as we already mentioned a few times, $p_{Y|X}(y|x)$ is undefined, but this case isn't a problem: no weighting is needed since an impossible outcome stays impossible.

To make things concrete, here is an example from the medical diagnosis problem where we observe $Y = \text{positive}$:

p_X

	healthy	infected
	0.999	0.001

$p_{Y|X}$

	healthy	infected
positive	0.01	0.99
negative	0.99	0.01

entry-wise multiply to get
unnormalized posterior

	healthy	infected
	0.00999	0.00099

We fix the fact that the unnormalized posterior table $\alpha(\cdot|y)$ isn't guaranteed to sum to 1 by renormalizing:

$$p_{X|Y}(x|y) = \frac{\alpha(x|y)}{\sum_{x'} \alpha(x'|y)} = \frac{p_X(x)p_{Y|X}(y|x)}{\sum_{x'} p_X(x')p_{Y|X}(y|x')}.$$

An important note: Some times we won't actually care about doing this second renormalization step because we will only be interested in what value that X takes on is more plausible relative to others; while we could always do the renormalization, if we just want to see which value of x yields the highest entry in the unnormalized table $\alpha(\cdot|y)$, we could find this value of x without renormalizing!

6.4 Maximum A Posteriori (MAP) Estimation

For a hidden random variable X that we are inferring, and given observation $Y = y$, we have been talking about computing the posterior distribution $p_{X|Y}(\cdot|y)$ using Bayes' rule. The posterior is a distribution for what we are inferring. Often times, we want to report which particular value of X actually achieves the highest posterior probability, i.e., the most probable value x that X can take on given that we have observed $Y = y$.

The value that X can take on that maximizes the posterior distribution is called the maximum a posteriori (MAP) estimate of X given $Y = y$. We denote the MAP estimate by $\hat{x}_{\text{MAP}}(y)$, where we make it clear that it depends on what the observed y is. Mathematically, we write

$$\hat{x}_{\text{MAP}}(y) = \arg \max_x p_{X|Y}(x|y).$$

Note that if we didn't include the "arg" before the "max", then we would just be finding the highest posterior probability rather than which value – or "argument" – x actually achieves the highest posterior probability.

In general, there could be ties, i.e., multiple values that X can take on are able to achieve the best possible posterior probability.

7 Independence Structure

7.1 Independent Events