Assignment 1: Bayes' Rule

Machine Learning

Fall 2019

Learning Objectives

- Gain familiarity with key ideas in ML (with a focus on probabilistic methods).
- Formal definition of a probability.
- Bayes' Rule
- Applications of Bayesian Inference

1 Motivation

The theme of uncertainty has cropped up a number of times so far this semester. For instance, we learned about the logistic regression model, which computes the probability that the output for a given input is 1. In this way, the logistic regression model explicitly represents the idea of the uncertainty of its predictions and uses the concept of probability to express these uncertainties. We've also seen examples of models that don't fit the data perfectly (well that's pretty much every model!). For example, the plot in Figure 1 shows a line of best fit that doesn't go through all of the data. The fit might be imperfect because we don't have the right model (a form of uncertainty) or because the thing we are trying to predict is inherently random in some way (another form of uncertainty).

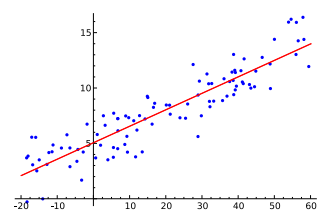


Figure 1: A dataset with one independent variable (x-axis) and one dependent variable. Also shown is the line of best fit.

It turns out that the notion of probability provides us with a powerful method to formalize the concept of uncertainty and allows us to reason about many types of uncertainty in a unified way. Formalizing uncertainty using probability theory will enable us to do a bunch of really cool things with respect to machine learning, including

- make explicit our assumptions about where data comes, the uncertainties present within it, and the impact of uncertainty on predictions based on that data.
- allow us to quantify our confidence in our model. For example, instead of returning the one best fitting model (as we did in the first module), we assign a probability to each possible model, and models that fit the data better will get a higher probability.
- give us a way to incorporate expert knowledge into our machine learning models. This expert knowledge will allow for us to learn more interpretable models.
- give us powerful tools to reason about fairness and bias in machine learning models.

2 Six Big Ideas in (Probabilistic) Machine Learning

Back by popular demand[1]... here are Six (more) Big ideas in ML! Here we're going to highlight ideas in ML that have specific connections to Bayesian machine learning.

[1] there was actually no specific demand for this, but we couldn't resist :).

Notice

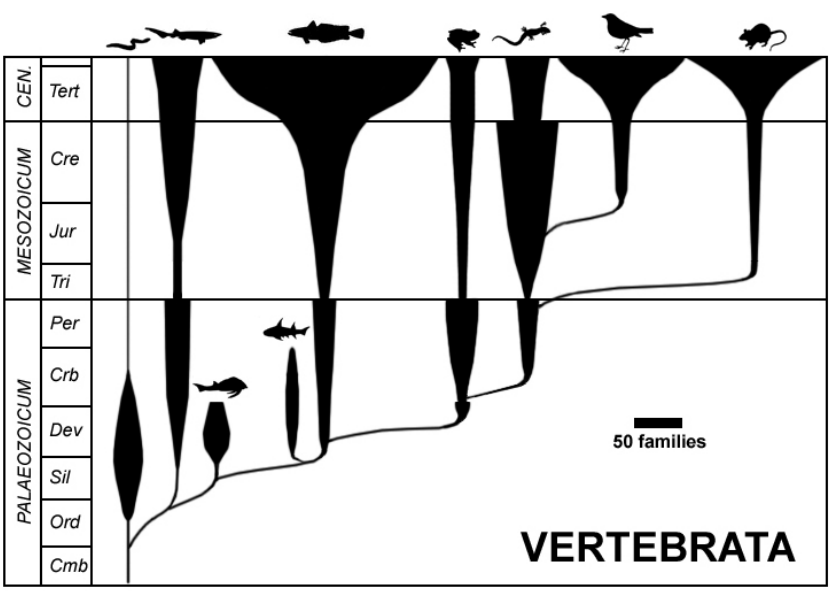
Our intent is to provide a larger framework for you to interpret the content you are learning. You should not expect to understand all of these ideas in any sort of detail. Please take these as a "wow isn't this cool / interesting," and avoid getting too lost in the weeds.

Exercise 1 (60 minutes)

Read the six big ideas in ML below. You will choose one of them and write a short response to it. Your response could incorporate something surprising you read, a thought-provoking question, your personal experience, an additional resource that builds upon or shifts the discussion. We hope that this reflection will help scaffold class discussions and get you thinking about your interests in the big space that is ML. Also, you have license from us to customize the structure of your response as you see fit. Aim for a response of a 1-2 paragraphs.

Idea 1: Bayesian Methods Can Illuminate Hidden Structures

One task in which Bayesian machine learning shines is in determining the hidden structures that underlie data. One very cool example of this is in the field of bioinformatics. A key idea in evolutionary biology is the notion of a phylogenetic tree. A phylogenetic tree captures the relationships between various entities (e.g., individual members of a species, different species, genes, etc.) in a hierarchy (or tree). For instance, the figure below shows a phylogenetic tree of the vertebrates.



Such a phylogenetic tree would be put together using a combination of fossil records along with analysis of modern day versions of these species. It turns out that Bayesian

machine learning is a key tool that is used to automatically infer phylogenetic trees from a wide variety of data sources. One example is using mitochondrial DNA collected from people all around the world and then using machine learning analysis to reconstruct the most likely phylogenetic tree that underlies these DNA samples. The result is an ancestral tree that helps reconstruct the ancient history of humans and snapshot of how they came to inhabit planet Earth. Reconstructions of this type have given support to the "out of Africa" theory of the origin of modern humans (namely that all people alive today can trace their lineage back to a group of people in sub-Saharan African from about 150,000 - 200,000 years ago).

Bayesian approaches to inferring phylogenetic trees have been used in a vast number of places. The abstract of A biologist's guide to Bayesian phylogenetic analysis describes it in the following way.

> Bayesian phylogenetic methods were introduced in the 1990s and have since revolutionised the way we analyse genomic sequence data3. Examples of such analyses include phylogeographic analysis of virus spread in humans inference of phylogeographic history and migration between species, analysis of species diversification rates, divergence time estimation, and inference of phylogenetic relationships among species or populations.

The concept of a phylogenetic tree has even been used used in linguistics as a mechanism to study the evolution of languages over time.

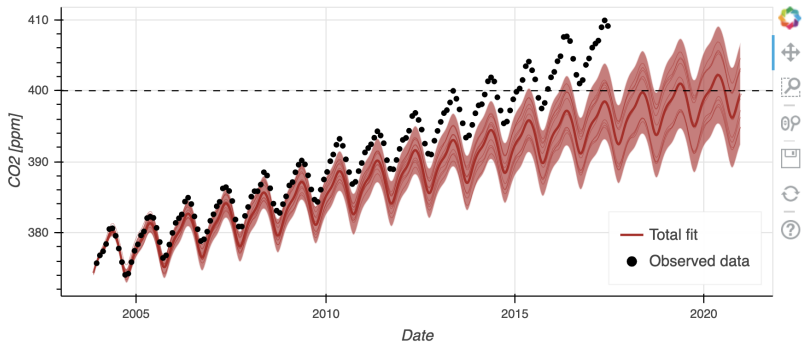
(If You're Interested): More examples of finding hidden structures in data Entropic Evidence for Linguistic Structure in the Indus Script

Idea 2: Bayesian Methods Allow Us to Write Probabilistic Programs

The key idea of Bayesian machine learning is that we can take causal models of how hidden structures give rise to observable data and invert them to allow us to use data to reason about the hidden structures that caused that data. One of the coolest ways in which this fundamental idea is being leveraged is in the field of probabilistic programming. In a probabilistic program, the programmer writes code to implement the causal model (the one that goes from hidden structures to data). In contrast to a typical program where we might take a particular input and produce a single output, a probabilistic program specifies a random process by which data is generated. Uncertainty is represented explicitly at each step in the program. Once the programmer has specified the causal model for how data is generated, the probabilistic programming framework can automatically infer the hidden structure given a sample of data!

For instance, one of the examples listed for the popular probabilistic programming library pymc3 shows how a model of CO2 concentration in the atmosphere could be created using a probabilistic program (see Mauna Loa CO2 example). The punchline graph from that notebook is reproduced below. In this case the programmer had to specify the process by which underlying phenomena (e.g., increases in CO2 levels globally, seasonal factors) give rise to observable data (CO2 measurements). This probabilistic program was then given atmospheric data up to 2004 and pymc3 automatically generated predictions for the CO2 data from 2004 onwards. In addition to the curve shown, the model was

able to decompose the curve into various components (including long range and seasonal effects).



(If You're Interested): Here are some more examples of probabilistic programmin Is computer vision really just inverse computer graphics? If you use probabilistic programming you can simply specify the forward model of how objects in a scene give rise to an image that we would capture on a camera. Probabilistic programming can then take an image and infer the underlying structure that gave rise to a test image. Examples of this include fitting a 3D models to human faces (follow the link to some cool videos) and predicting how a geometric scene would look from an arbitrary vantage point from a single training image.

Idea 3: Bayesian Methods Can Help Us Understand Learning and Perception in Humans and Robots

One common challenge that humans and robots (and other animals for that matter) face is the need to interpret ambiguous information in the world around them. For instance, the photons of light that hit our retinas create a 2D map of a 3D world. Despite the fact that there are many possible 3D worlds that could have created the same pattern of light on our retinas, our brain, for the most part, has no trouble figuring out the correct 3D structure of the world[2].

[2] optical illusions are a good example of when this does not hold (i.e., our brain is tricked into inferring the wrong 3D structure).

One approach that scientists have taken to understand how humans accurately extract structure in the face of ambiguous information is by applying what is known as the computational approach. In the computational approach, one formalizes the goal of the agent as well as the constraints facing it. For example, perhaps a person is trying to accurately understand the 3D structure of their environment but is constrained to only have access to a 2D projection of that environment (i.e., the 2D projection is what is captured on our retina). Once we have formalized the problem facing the agent, we can then use computational tools to compute the optimal solution to the problem and compare this solution to what people actually do. Through this comparison we may learn more about the nature of human cognition and the reasons why humans think and act the way they do. In David Marr's classic work "Vision," he states the core idea in the following way.

> Trying to understand perception by studying only neurons is like trying to understand bird flight by studying only feathers: It just cannot be done. In order to understand bird

> flight, we have to understand aerodynamics; only then do the structure of feathers and the different shapes of birds' wings make sense. [Marr, 1982] p. 27

The science of "aerodynamics" is being invoked in the quote to reference a fundamental theory that governs all systems that might try to fly. When it comes to cognition, the idea of Bayesian inference, which will form the basis of Bayesian machine learning, has increasingly emerged as the "aerodynamics" of thought and reasoning. Scientists and engineers have used Bayesian approaches to understand a vast array of human cognition and behavior.

- Sensory perception
- Motor control
- Babies can use Bayesian inference to understand physics... sort of
- The tendency of people to with divergent beliefs to become more polarized when presented with the same evidence (think current politics)
- How people behave in a simulated gambling environment
- How people ascribe trust to others
- How people acquire language

Idea 4: Bayesian Methods Give Us a Language for Reasoning about Algorithmic Fairness

We've talked about bias in machine learning models (e.g., when we discussed Gender Shades). The examples given in the Gender Shades paper were pretty cut-and-dry — the gender classification models that many companies have released work very poorly on women and people of color (and especially the intersection of the two). In other situations it is not so straightforward to reason about bias. For instance, in the well-known ProPublica piece Machine Bias the authors describe how a tool for predicting criminal recidivism (whether a criminal would reoffend) was biased against African Americans. This piece was one of the most important in the field of FAT-ML in that it sparked some work on articulating various definitions of algorithmic fairness. Most of these models of fairness are articulated using the language of probability. Further, some of the same formalisms that have been used to develop probabilistic machine learning approaches can be used to understand and reason about bias in an algorithm.

We'll be studying various notions of algorithmic fairness, but if you're interested an in-depth overview of this is provided in Fairness criteria through the lens of directed acyclic graphical models. It's also worth mentioning that many of these notions of fairness have also been challenged on the grounds that they are overly abstract and gloss over important context in determining whether something is biased or not.

Idea 5: Bayesian Methods Allow for Learning More Interpretable and Flexible Models

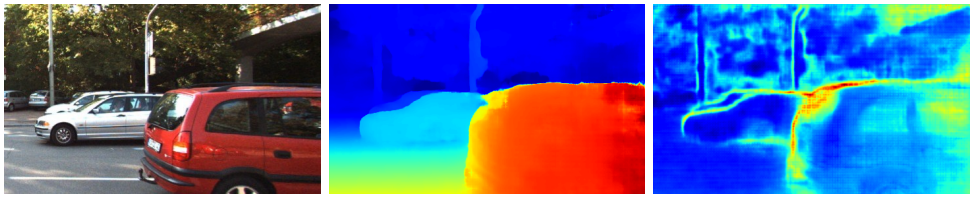
Since Bayesian models typically start from a causal model of how observable data are generated, when this process is inverted, you are left with an understanding of the most probable causes of the data you observed. Oftentimes this output is easier to interpret than what might be generated by, say, a neural network, which has no explicit causal model of data generation. As a motivating example, take for example the usage of computers to help doctors diagnose patients (this is sometimes called a Clinical Decision Support System). Here are two ways in which we might tackle this problem using neural networks versus Bayesian machine learning.

- Neural networks: we would get a dataset of tons of patients, their symptoms, and their diagnoses. We would train a neural network model to map from symptoms to diagnoses.
- We would have an explicit model of how underlying factors (e.g., diseases, age, environmental conditions) give rise to symptoms. We would fit some of the parameters of that model using clinical data or through prior knowledge (e.g., the intuition of doctors). When we wanted to diagnose a patient, we would use Bayesian inference to predict the likely underlying disease that contributed to the patient's symptoms.

The potential advantages of the Bayesian approach are that they way it makes decisions might be more understandable / familiar to a doctor (since it is not that different from how a doctor might diagnose patients themselves). Further, if something changed about some aspect of the environment (e.g., the sensitivity of a test for a disease increased or the prevalence of a disease decreased), the predictions would naturally adjust. It is less easy to see how to incorporate these sorts of changes into the neural network approach.

This is not to bash neural networks! There is even a burgeoning field designed to bring some of the advantages of Bayesian approaches to the field of deep learning. A figure from this work is shown below. The network in question is able to represent not only its prediction regarding the likely depth map of given a stereo image, but also the uncertainty that the model has for various parts of the image.

Figure 2: Bayes (pictured) meets neural networks. Generated using deepdreamgenerator.com



Idea 6: Bayesian Methods Pose Difficult Computational Challenges

While we've painted a fairly rosy picture of Bayesian methods, but they are certainly not without their challenges. One of the biggest challenges is computational (see here for a summary of current computational challenges in Bayesian learning). The key reason

that Bayesian methods can be computationally expensive is that they involve solving the following sorts of problems.

- Bayesian methods involve inverting causal models of the world. This inversion process may not be well-posed and could result in very inefficient computations.
- Bayesian methods involve representing uncertainty over large model spaces. Just representing this model space, let alone using it for making predictions can be hard.

Therefore, while Bayesian methods are very attractive (and will pose no significant computational problems in some important special cases, which we will learn about in this module), efficiently using Bayesian methods is an active area of research.

3 Probability

Hopefully the previous section left you feeling excited to learn more about the theory that underlies these big ideas. Next, we'll take our first steps towards learning this theory.

3.1 Intuition

Most of us are used to thinking that events can be probabilistic, that is we can attach some probability to whether or not they occur. Take for example flipping a coin. We could think of the event that the coin comes up heads as having probability 0.5. That is, there is an even chance that it happens versus doesn't happen. Further, we can say that an event is *observable* if we are able to directly observe whether it occurred. For instance, whether a coin comes up heads is an observable event since you can ultimately observe the outcome of the flip. In contrast, some events would be considered unobservable if they are unable to be directly ascertained by human senses. A classic example of this would be whether a scientific theory is true or not. It is impossible to directly observe whether the theory is true, but you might be able to observe events that are consistent or inconsistent with the theory.

Exercise 2 (10 minutes)

Come up with some examples (approximately 3 is a good number to go for) of observable events that are probabilistic in nature. For each event, provide the probability that it occurs or explain what factors would determine the probability. Some potential ideas to get you going: sporting events, elections, weather, etc.

3.2 Formal Definition

Next, we'll define more formally[3] what we mean by a probability. Having this formal definition will give us the ability to determine useful rules for manipulating and reasoning about probabilities. To define the concept of a probability, we'll need to specify two ingredients.

[3] this is not the full definition of a probability space used in modern mathematics. For the purposes of most people that *use* probability theory on actual problems, the full definition is needlessly complex. For instance, as a grad student I (Paul) never saw the full definition in any of my ML courses. Use the following link if you want a more in-depth discussion of the parts of the formal definition that are tricky (our expectation is that you won't want this discussion!).

3.3 Events

An *event* is something that may or may not occur in response to some random process. For instance, we could define the event that a coin comes up heads when it is flipped. We often use capital letters to indicate events. Since we've been using capital letters to also represent matrices, in our materials we'll use a cool mathy-looking calligraphic font to represent events. For instance, we might use the symbol $\mathcal{H}$ to refer to the event that a coin flip comes up heads. It's important to emphasize that a single random process can have many associated events. For instance, for the coin flip example we might also define $\mathcal{T}$ to be the event that the coin comes up tails (or $\mathcal{U}$ to indicate that event that the coin rotated at least 10 times in the air when we flipped it).

Further, events don't necessarily have to be mutually exclusive. For instance, we might define the event $\mathcal{R}_h$ to indicate the event that the Republican party controls the majority in the House of Representatives following the 2020 election and $\mathcal{D}_s$ to indicate the event that the Democratic party controls the majority in the Senate following the 2020 election. Both (or none) of these events could occur.

3.4 Probability Measure Function

The probability measure function assigns a probability to the occurrence of any particular event. We can think of this probability measure function as taking as input an event and outputting a probability. For instance, $p(\mathcal{E})$ provides the probability that event $\mathcal{E}$ occurs according to probability measure p. All probability measure functions must satisfy the following properties.

- $0 \leq p(\mathcal{E}) \leq 1$: the probability of an event ranges from 0 (an impossible event) to 1 (an event that will always occur).
- Given a set of n events $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_n$ that are disjoint (i.e., no two can occur simultaneously)

$$p(\mathcal{E}_1 \text{ or } \mathcal{E}_2 \text{ or } \ldots \text{ or } \mathcal{E}_n) = \sum_{i=1}^{n} p(\mathcal{E}_i) \ . \tag{1}$$

  The equation above specifies what's sometimes called the union rule of probability. It states that the probability of one of these disjoint events occurring must be equal to the sum of the probability of each of the events occurring. You will also sometimes see Equation 1 written as

$$p(\mathcal{E}_1 \cup \mathcal{E}_2 \cup \ldots \cup \mathcal{E}_n) = \sum_{i=1}^{n} p(\mathcal{E}_i) \ . \tag{2}$$

  If you're not familiar with the symbol $\cup$ it is the symbol for a union of two sets. The reason you'll sometimes see this notation is that in the full definition of a probability space, an event is defined as a set (as stated in the margin note earlier in this section, you need not worry about the most rigorous definition in this course).

- Given a set of (not necessarily disjoint) events $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_n$ where at least one of these n events must occur

$$p(\mathcal{E}_1 \text{ or } \mathcal{E}_2 \text{ or } \ldots \text{ or } \mathcal{E}_n) = 1 \ . \tag{3}$$

This rule just states that if we have an exhaustive set of events (that cover all possible cases), at least one of them *must* occur.

3.5 Complement Rule for Probability

Given the definition of probability detailed above, it follows that if the probability of an event happening is $p(\mathcal{E})$ then the probability of the event *NOT* happening is $1 - p(\mathcal{E})$. The following are common ways of to express this relationship (we'll use Equation 4 in this class). These all say the same thing (the only difference is notation).

$$\begin{aligned} p(\neg\mathcal{E}) &= 1 - p(\mathcal{E}) \\ p(\text{not } \mathcal{E}) &= 1 - p(\mathcal{E}) \\ p(\overline{\mathcal{E}}) &= 1 - p(\mathcal{E}) \\ p(\mathcal{E}') &= 1 - p(\mathcal{E}) \end{aligned} \tag{4}$$

We point out these alternate notations not to confuse you (we'd never do that!) but to help you interpret various external resources you might find on these topics.

Exercise 3 (10 minutes)

Here are some diagnostic questions to make sure that you got the basic ideas.

(a) Suppose $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3$ are disjoint events. Further, suppose that one of these events must occur. Which of the following functions are valid probability measure functions?

$$p_1(\mathcal{E}_1) = \frac{1}{10}, p_1(\mathcal{E}_2) = \frac{1}{5}, p_1(\mathcal{E}_3) = \frac{7}{10}$$

$$p_2(\mathcal{E}_1) = \frac{11}{10}, p_2(\mathcal{E}_2) = \frac{-1}{10}, p_2(\mathcal{E}_3) = 0$$

$$p_3(\mathcal{E}_1) = \frac{1}{10}, p_3(\mathcal{E}_2) = \frac{1}{5}, p_3(\mathcal{E}_3) = \frac{1}{2}$$

$$p_4(\mathcal{E}_1) = 1, p_4(\mathcal{E}_2) = 0, p_4(\mathcal{E}_3) = 0$$

(b) The Birthday Problem is a well-known probability problem often used in discrete math courses. According to the Wikipedia article on the Birthday Problem, the probability that at least two students among the 70 students in machine learning this semester share the same birthday is 0.999. What is the probability that no two students share the same birthday?

4 *Bayes' Rule*

External Resource(s) (60 minutes)

Learning Objectives

Note that these learning objectives have been written to be very specific (based on feedback from the course survey). When you first read them, you probably won't know what they mean in detail. As you go through the readings, hopefully the more precise statement of these learning objectives will be useful for assessing your understanding of the provided resources.

- When Bayes' rule is useful (i.e., when $p(A|B)$ is easier to work with than $p(B|A)$).
- The idea of a conjoint probability $p(\mathcal{A}, \mathcal{B})$ (note: alternate notations include $p(\mathcal{A}$ and $\mathcal{B})$ and $p(\mathcal{A} \cap \mathcal{B})$).
- The definition of a conditional probability $p(\mathcal{A}|\mathcal{B}) = \frac{p(\mathcal{A},\mathcal{B})}{p(\mathcal{B})}$.
- The equation for the product rule $p(\mathcal{A}, \mathcal{B}) = p(\mathcal{B})p(\mathcal{A}|\mathcal{B}) = p(\mathcal{A})p(\mathcal{B}|\mathcal{A})$ (Allen calls this the probability of a conjunction).
- The equation for Bayes' rule $p(\mathcal{A}|\mathcal{B}) = \frac{p(\mathcal{B}|\mathcal{A})p(\mathcal{A})}{p(\mathcal{B})}$.

Allen Downey (ever heard of him?) wrote an excellent book called Think Bayes that introduces Bayesian analysis. The first chapter (which you should read) starts with a less formal definition of probability than the one we gave earlier. The chapter then gives intuitions around conjoint probability (the probability that multiple events occur simultaneously), conditional probability (the probability that some event occurs conditioned on another event having occurred), and finally to Bayes' rule (a surprisingly easy theorem to derive that allows you to write one conditional probability distribution in terms of another). The Monty Hall problem in section 1.7 is probably okay to gloss over (see Allen's note at the end of that section for why this is the case).

Allen's treatment of the material is, of course, not the only one out there (we like it for its focus on building intuition and focusing on the key ideas). Here are

some other resources you might consider checking out (they are optional).

- Khan Academy Video on Bayes' Theorem shows some simple applications of Bayes' rule and explains why it is a convenient way to reason about the probability of hypothesis given data).
- Veritasium Episode on Bayes' Theorem has a bit more history and philosophy of Bayes' Theorem along with some nice visualizations. It also includes the presenter walking on a very scenic mountain (for some reason), so there's that if nothing else.
- I (Paul) ran across this example of applying Bayes' rule to a real world problem. It was created by a grad school friend of mine and is hilarious (lots of Cat Memes). I did notice that there is a mistake in the math at the 8:12 mark in the video (he states that $p(\text{alarm}|\text{no theft}) = 1 - p(\text{alarm}|\text{theft})$, which is not necessarily the case). It's still a good video though.

Exercise 4 (20 minutes)

Notice

If you are having trouble dealing with the denominator that you get when you apply Bayes' rule, you may want to skip ahead to the section on the *Marginalization Rule for Probabilities* and then return to these problems.

(a) You are given three coins. Two are of type 1, we'll call this type $\mathcal{C}_1$, and one is of type 2, we'll call this type $\mathcal{C}_2$. If you flip a coin of type 1, the coin will come up heads with probability $\frac{4}{5}$ (i.e., $p(\mathcal{H}|\mathcal{C}_1) = \frac{4}{5}$). If you flip a coin of type 2, the coin will come up heads with probability $\frac{1}{2}$ (i.e., $p(\mathcal{H}|\mathcal{C}_2) = \frac{1}{2}$). Suppose you choose one of the three coins (there is no way for you to tell them apart), flip it once, and it comes up heads. What is the probability that you flipped a coin of type 2 (i.e., what is $p(\mathcal{C}_2|\mathcal{H})$)? After you compute your answer, compare it with the probability that a randomly selected coin was of type 2. Does the relationship between this prior probability and the posterior probability make sense?

(b) You train a neural network to identify whether an image contains a picture of a Chihuahua or a Blueberry Muffin (you know you want to click the link!). Let's further assume that there are no images that contain both a muffin and a Chihuahua (of course as we all know Chihuahuas love muffins). Based on your project report for module 1, you know that if the image contains a Chihuahua, your model will identify it as such with probability 0.9. Also, you know that if the model contains a muffin, your model will identify it as such with probability 0.8. You now decide to test your model on a dataset which contains 80%

muffins. Assuming that your model predicted that an image contained a muffin, what is the probability that it actually contains a muffin? (you can assume that the performance of your model doesn't change when run on this new dataset).

5 *Marginalization Rule for Probabilities*

The application of Bayes' rule often proceeded according to the following outline. First, we would define an event we want to reason about. For instance, we might define $\mathcal{D}$ as the event that a person has a disease and $\mathcal{S}$ as the event that a particular symptom is observed. If we want to know $p(\mathcal{D}|\mathcal{S})$ we apply Bayes' rule like so.

$$p(\mathcal{D}|\mathcal{S}) = \frac{p(\mathcal{S}|\mathcal{D})p(\mathcal{D})}{p(\mathcal{S})} \tag{5}$$

In order to calculate $p(\mathcal{S})$, some of the resources simply gave a number (e.g., in the Khan Academy video the premise was that you Googled to find this value), used a convenient trick to get it (as in Allen's M&M example), or used the following calculation (as in the Veritasium and the Car Alarm videos).

$$p(\mathcal{S}) = p(\mathcal{D})p(\mathcal{S}|\mathcal{D}) + p(\neg\mathcal{D})p(\mathcal{S}|\neg\mathcal{D}) \tag{6}$$

We wanted to revisit this calculation as it is hiding away some pretty powerful and interesting stuff. This calculation can be derived using the technique of marginalizing a probability measure function. The basic motivation for this technique is that sometimes you'd like to compute the probability of some event, $\mathcal{A}$, but it is difficult to do so directly. Instead you can introduce another event, $\mathcal{B}$, and write $p(\mathcal{A})$ as:

$$p(\mathcal{A}) = p(\mathcal{A}, \mathcal{B}) + p(\mathcal{A}, \neg\mathcal{B}) \tag{7}$$

In the equation above we sometimes say that we are *marginalizing out* $\mathcal{B}$ (by summing over the two possibilities: that $\mathcal{B}$ occurred and that $\mathcal{B}$ did not occur).

Exercise 5 (15 minutes)

Using Equation 7 and the product rule of probability, show that Equation 5 is true.

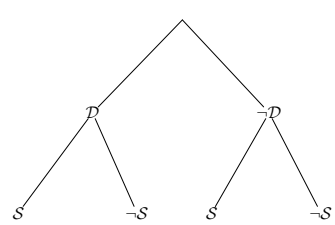
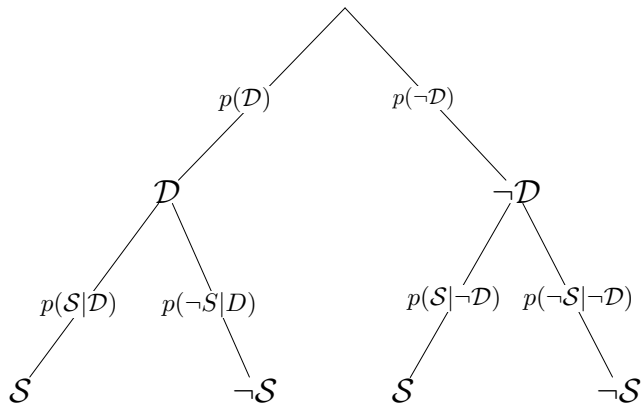


Figure 3: A tree diagram of the events $\mathcal{D}$ (has a disease) and $\mathcal{S}$ (has a symptom).

Another way to think about marginalization is to draw a tree where you have the event, which you are marginalizing out ($\mathcal{D}$ in the previous exercise) at the first level of the tree and the variable you want to know the probability of ($\mathcal{S}$ in the previous exercise) at the next junction in the tree (see Figure 3).

Further, we annotate the arrows with the conditional probability of the event conditioned on the things further up in the tree (note that for $\mathcal{D}$ there is nothing farther up the tree, so we just write $p(\mathcal{D})$ or $p(\neg\mathcal{D})$).



If you want to find the probability of a joint probability (e.g., $p(\mathcal{D}, \neg\mathcal{S})$), you can follow the corresponding path, multiplying probabilities as you go. For example, from examining the graph above we get

$$p(\mathcal{D}, \neg\mathcal{S}) = p(\mathcal{D})p(\neg\mathcal{S}|\mathcal{D}) \ . \tag{8}$$

A marginal probability for an event (e.g., $p(\mathcal{S})$) can be found by summing over all paths that arrive at the event. For example, from examining the graph above we see that there are two paths to $\mathcal{S}$, which allows us to compute $p(\mathcal{S})$ as

$$p(\mathcal{S}) = p(\mathcal{D})p(\mathcal{S}|\mathcal{D}) + p(\neg\mathcal{D})p(\mathcal{S}|\neg\mathcal{D}) \ . \tag{9}$$

Exercise 6 (20 minutes)

Do problem 2 from this assignment. They use $\mathcal{E}'$ to refer to the event $\neg\mathcal{E}$.

6 *Random Variables*

We've talked about the concept of an event that captures whether something happens as a result of some random process. It turns out that it is very useful in machine learning and probabilistic modeling to talk about a variable that captures some quantity that is a result of some random process. We call this entity a *random variable*.

External Resource(s) (20 minutes)

Watch the following two videos from Khan Academy on Random Variables.

- Khan Academy Video on Random Variables.
- Discrete and Continuous Random Variables (note: for now we are interested in discrete random variables).

Now that you have a sense of what a random variable is, we'll introduce the notion of probability mass function (or PMF). A PMF is a function that assigns a probability to a discrete random variable taking on a particular value as a result of a random process.

We'll use capital, unbolded letters to refer to random variables (e.g., the random variable X) and $p(X = k)$ to refer to the probability that X takes on value k).

For example, if we were flipping a fair coin 3 times, we might define a random variable X as follows.

$$X = \text{the number of coins that come up heads in 3 flips} \tag{10}$$

The probability mass function (PMF) is then defined over all possible values that X could possibly take on. If don't know how we arrived at these values, that is fine. These are results that can be derived using basic combinatorics (go take Sarah Spence Adams' class and learn about how to do this).

$$\begin{aligned}
p(X = 0) &= p(0 \text{ heads come up}) = \frac{1}{8} \\
p(X = 1) &= p(1 \text{ heads come up}) = \frac{3}{8} \\
p(X = 2) &= p(2 \text{ heads come up}) = \frac{3}{8} \\
p(X = 3) &= p(3 \text{ heads come up}) = \frac{1}{8}
\end{aligned}$$

We don't have an exercise for you to do here, but it is important for you to understand what a PMF is when digging into the companion notebook.

7 *Basic Bayes in Python*

External Resource(s) (30 minutes)

Go through the assignment 1 companion notebook.

Open in Colab

Assignment 1 Companion Notebook

In this notebook you'll be applying Bayesian analysis to a couple of problems. The presentation will follow Allen's workshop from PyCon Bayes Made Simple.

First, install some packages and import the necessary libraries.

In [1]:

```
!pip install empyrical-dist

%matplotlib inline

import numpy as np
import pandas as pd

import seaborn as sns
sns.set_style('white')
sns.set_context('talk')

import matplotlib.pyplot as plt

from empyrical_dist import Pmf
```

```
Collecting empyrical-dist
  Downloading
https://files.pythonhosted.org/packages/33/01/677b7cfba73873b91d6239ae961870028469f9b2c85e5830b4d20
f1f/empyrical_dist-0.2.0.tar.gz
Building wheels for collected packages: empyrical-dist
  Building wheel for empyrical-dist (setup.py) ... done
  Created wheel for empyrical-dist: filename=empyrical_dist-0.2.0-cp36-none-any.whl size=8074
sha256=d24acf9f5d2ec5003a6e27510b6cc8693579132bcd75c60e13126e04055bfbd9
  Stored in directory:
/root/.cache/pip/wheels/b5/33/56/c1f9f4c185160b176514f3d2550eabaf9a2cd514ec033ebba6
Successfully built empyrical-dist
Installing collected packages: empyrical-dist
Successfully installed empyrical-dist-0.2.0
```

Working with Pmfs

Recall from the main assignment document that a PMF (probability mass function) specifies the probability that a random variable takes on a particular value. In this problem we can think of the random variable X as representing the outcome of rolling a single, six-sided die.

Create a Pmf object to represent a six-sided die.

In [0]:

```
d6 = Pmf()
```

A Pmf is a map from possible outcomes to their probabilities.

In [0]:

```
for x in [1,2,3,4,5,6]:
    d6[x] = 1
```

Initially the probabilities don't add up to 1. *Just as a probability measure must add to 1 when summed over a set of mutually exclusive, exhaustive events, a PMF must also add to 1*

In [4]:

```
d6
```

Out[4]:

	probs
1	1
2	1
3	1
4	1
5	1
6	1

`normalize` adds up the probabilities and divides through. The return value is the total probability before normalizing.

In [5]:

```
d6.normalize()
```

Out[5]:

6

Now the Pmf is normalized.

In [6]:

```
d6
```

Out[6]:

	probs
1	0.166667
2	0.166667
3	0.166667
4	0.166667
5	0.166667
6	0.166667

And we can compute its mean (which only works if it's normalized).

In [7]:

```
d6.mean()
```

Out[7]:

3.5

The cookie problem

The setup of this problem is in the chapter 1 reading from Think Bayes. The relevant part is that you have two bowls of cookies. Bowl 1 contains 3/4 vanilla cookies and 1/4 chocolate and bowl 2 contains 1/2 vanilla cookies and 1/2 chocolate.

`Pmf.from_seq` makes a `Pmf` object from a sequence of values.

Here's how we can use it to create a `Pmf` with two equally likely hypotheses.

In [11]:

```
cookie = Pmf.from_seq(['Bowl 1', 'Bowl 2'])
cookie
```

Out[11]:

	probs
Bowl 1	0.5
Bowl 2	0.5

Now we can update each hypothesis with the likelihood of the data (a vanilla cookie). *This is Bayes' rule in action!*

In [12]:

```
cookie['Bowl 1'] *= 0.75
cookie['Bowl 2'] *= 0.5
cookie.normalize()
```

Out[12]:

```
0.625
```

And display the posterior probabilities.

In [13]:

```
cookie
```

Out[13]:

	probs
Bowl 1	0.6
Bowl 2	0.4

You might be wondering what happened to the denominator in Bayes' rule. You would think that to update the probability of bowl 1 you should be doing the following calculation:

$$p(\text{Bowl 1} \mid \text{vanilla}) = \frac{p(\text{vanilla} \mid \text{Bowl 1}) p(\text{Bowl 1})}{p(\text{vanilla})}$$

.

in Allen's code, all he does is multiply the prior by the likelihood (the numerator of the previous equation). It turns out that applying the *normalize* function will have the same effect as dividing by $p(\text{vanilla})$. You can take this on faith, or show why this is true yourself (it should be pretty straightforward, post on NB if you need help).

Notebook Exercise 1

Suppose we put the first cookie back, stir, choose again from the same bowl, and get a chocolate cookie.

What are the posterior probabilities after the second cookie?

Hint: The posterior (after the first cookie) becomes the prior (before the second cookie).

The dice problem

Create a Suite to represent dice with different numbers of sides (in this case 4, 6, 8, and 12).

In [15]:

```
dice = Pmf.from_seq([4, 6, 8, 12])
dice
```

Out[15]:

Out[19]:

	probs
4	0.25
6	0.25
8	0.25
12	0.25

Notebook Exercise 2

We'll solve this problem two ways. First we'll do it "by hand", as we did with the cookie problem; that is, we'll multiply each hypothesis by the likelihood of the data, and then renormalize.

Update `dice` based on the likelihood of the data (rolling a 6), then normalize and display the results.

Notebook Exercise 3

Now let's do the same calculation using `Pmf.update`, which encodes the structure of a Bayesian update.

Define a function called `likelihood_dice` that takes `data` and `hypo` and returns the probability of the data (the outcome of rolling the die) for a given hypothesis (number of sides on the die).

Hint: What should you do if the outcome exceeds the hypothetical number of sides on the die?

Here's an outline to get you started.

In [0]:

```
def likelihood_dice(data, hypo):
    """Likelihood function for the dice problem.

    data: outcome of the die roll
    hypo: number of sides

    returns: float probability
    """
    # TODO: fill this in!
    return 1
```

More Fun with Dice

If we get more data, we can perform more updates.

In [0]:

```
for roll in [8, 7, 7, 5, 4]:
    dice.update(likelihood_dice, roll)
```

Here are the results.

In [22]:

```
dice
```

Out[22]:

	probs
4	0.000000
6	0.000000
8	0.919294
12	0.080706

The German tank problem

The German tank problem is actually identical to the dice problem. This one is a cool example, because it was actually used to estimate German monthly tank production during WWII. Here are the results of the analysis.

	Month	Statistical estimate	Intelligence estimate	German records
June 1940	169	1,000	122	
June 1941	244	1,550	271	
August 1942	327	1,550	342	

In [0]:

```
def likelihood_tank(data, hypo):
    """Likelihood function for the tank problem.

    data: observed serial number
    hypo: number of tanks

    returns: float probability
    """
    if data > hypo:
        return 0
    else:
        return 1 / hypo
```

Here is the update after seeing Tank #42.

In [24]:

```
tank = Pmf.from_seq(range(100))
tank.update(likelihood_tank, 42)
tank.mean()
```

Out[24]:

```
66.32784309363326
```

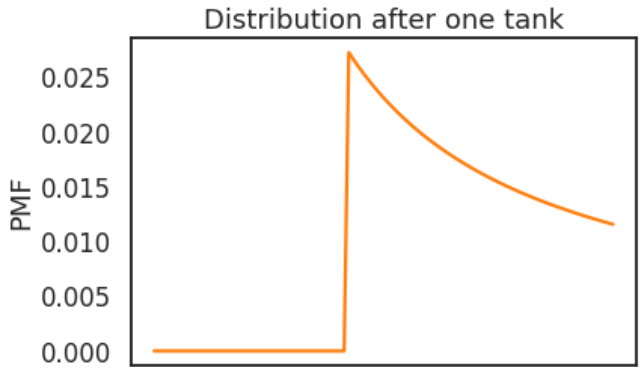
And here's what the posterior distribution looks like.

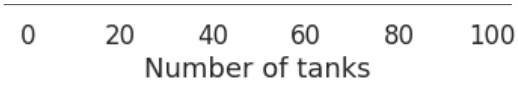
In [27]:

```
def decorate_tank(title):
    """Labels the axes.

    title: string
    """
    plt.xlabel('Number of tanks')
    plt.ylabel('PMF')
    plt.title(title)

tank.plot()
decorate_tank('Distribution after one tank')
```





Notebook Exercise 4

Suppose we see another tank with serial number 17. What effect does this have on the posterior probabilities?

Update the `Pmf` with the new data and plot the results.