Joint Probability

Many interesting problems involve not one random variable, but rather several interacting with one another. In order to create interesting probabilistic *models* and to reason in real world situations, we are going to need to learn how to consider several random variables jointly.

In this section we are going to use disease prediction as a working example to introduce you to the concepts involved in probabilistic models. The general question is: a person has a set of observed symptops. Given the symptoms what is the probability over each possible disease?

We have already considered events that co-occur and covered concepts such as independence and conditional probability. What is new about this section is (1) we are going to cover how to handle *random variables* which co-occur and (2) we are going to talk about how computers can reason under large probabilistic models.

1. Joint Probability Functions

For single random variables, the most important information was the PMF or, if the variable was continuous, the PDF. When dealing with two or more variables, the equivalent function is called the Joint function. For discrete random variables, it is a function which takes in a value for each variable and returns the probability (or probability density for continuous variables) that each variable takes on its value. For example if you had two *discrete* variables the Joint function is:

$$\mathrm{P}(X = x, Y = y) \qquad \text{Joint function for } X \text{ and } Y$$

You should read the comma as an "and" and as such this is saying the probability that $X = x$ and $Y = y$. Again like for single variables, as shorthand, we often write just the values and it implies that we are talking about the probability of the random variables taking on those values. This notation is convenient because it is shorter, and it makes it explicit that the function is operating over two parameters. It requires to to recall that the event is a *random variable* taking on the given value.

$$\mathrm{P}(x, y) \qquad \text{Shorthand for } \mathrm{P}(X = x, Y = y)$$

If any of the variables are continuous we use different notation to make it clear that we need a probability density function, something we can integrate over to get a probability. We will cover this in detail:

$$f(X = x, Y = y) \qquad \text{Joint density function if } X \text{ or } Y \text{ are continuous}$$

The same idea extends to as many variables as you have in your model. For example if you had three discrete random variables X, Y, and Z, the joint probability function would state the likelihood of an assignment to all three: $\mathrm{P}(X = x, Y = y, Z = z)$.

2. Joint Probability Tables

> ***Definition***: Joint Probability Table
> A joint probability table is a way of specifying the "joint" distribution between multiple random variables. It does so by keeping a multi-dimensional lookup table (one dimension per variable) so that the probability mass of any assignment, eg $\mathrm{P}(X = x, Y = y, \ldots)$, can be directly looked up.

Let us start with an example. In 2020 the Covid-19 pandemic disrupted lives around the world. Many people were unable to get tested and had to determine whether or not they were sick based on home diagnosis. Lets build a very simple probabilistic model to enable us to make a tool which can predict the probability of having the illness given observed symptoms. To make it clear that this is a pedagogical example, lets consider a made up illness called Determinitis. The two main symptoms are fever and loss of smell.

Variable	Symbol	Type
Has Determinitis	D	Bernoulli (1 indicates has Determinitis)
Fever	F	Categorical (none, low, high)
Can Smell	S	Bernoulli (1 indicates can smell)

A joint probability table is a brute force way to store the probability mass of a particular assignment of values to our variables. Here is a probabilistic model for our three random variables (aside: the values in this joint are realistic and based on reasearch, but are primarily for teaching. Consult a doctor before making medical decisions).

$D = 0$

	$S = 0$	$S = 1$
$F = \text{none}$	0.024	0.783
$F = \text{low}$	0.003	0.092
$F = \text{high}$	0.001	0.046

$D = 1$

	$S = 0$	$S = 1$
$F = \text{none}$	0.006	0.014
$F = \text{low}$	0.005	0.011
$F = \text{high}$	0.004	0.011

A few key observations:

- Each cell in this table represents the probability of one assignment of variables. For example the probability that someone cant smell, $S = 0$, has a low fever, $F = \text{low}$, and has the illness, $D = 1$, can be directly read off the table: $P(D = 1, S = 0, F = \text{low}) = 0.005$.
- These are joint probabilities *not* conditional probabilities. The value 0.005 is the value of illness, no smell and low fever. It is not the probability of no smell and low fever given illness. A table which stored conditional probabilities would be called a conditional probability table, this is a joint probability table.
- If you sum over all cells, the total will be 1. Each cell is a mutually exclusive combination of events and the cells are meant to span the entire space of possible outcomes.
- This table is large! We can count the number of cells using the step rule of counting. If n_i is the number of different values that random variable i can take on, the number of cells in the joint table is $\prod_i n_i$.

3. Properties of Joint Distributions

There are many properties of a random variable of any random variable some of which we will dive into extensively. Here is a brief summary. Each random variable has:

Property	Notation Example	Description
Distribution Function (PMF or PDF)	$\text{P}(X = x, Y = y, \ldots)$ or $f(X = x, Y = y, \ldots)$	A function which maps values the RV can take on to likelihood.
Cumulative Distribution Function (CDF)	$F(X < x, Y < y, \ldots)$	Probability that each variable is less than its corresponding parameter
Covariance	$\sigma_{X,Y}$	A measure of how much two random variables vary together.
Correlation	$\rho_{X,Y}$	Normalized co-variance

Multinomial

The multinomial is an example of a *parametric* distribution for multiple random variables.

Say you perform n independent trials of an experiment where each trial results in one of m outcomes, with respective probabilities: $p_1, p_2, \ldots, p_m$ (constrained so that $\sum_i p_i = 1$). Define X_i to be the number of trials with outcome i. A multinomial distribution is a closed form function that answers the question: What is the probability that there are c_i trials with outcome i. Mathematically:

$$P(X_1 = c_1, X_2 = c_2, \ldots, X_m = c_m) = \binom{n}{c_1, c_2, \ldots, c_m} \cdot p_1^{c_1} \cdot p_2^{c_2} \cdots p_m^{c_m}$$

Often people will use the product notation to write the exact same equation:

$$P(X_1 = c_1, X_2 = c_2, \ldots, X_m = c_m) = \binom{n}{c_1, c_2, \ldots, c_m} \cdot \prod_i p_i^{c_i}$$

Example: A 6-sided die is rolled 7 times. What is the probability that you roll: 1 one, 1 two, 0 threes, 2 fours, 0 fives, 3 sixes (disregarding order).

$$\begin{aligned} &P(X_1 = 1, X_2 = 1, X_3 = 0, X_4 = 2, X_5 = 0, X_6 = 3) \\ &\quad = \frac{7!}{2!3!}\left(\frac{1}{6}\right)^1\left(\frac{1}{6}\right)^1\left(\frac{1}{6}\right)^0\left(\frac{1}{6}\right)^2\left(\frac{1}{6}\right)^0\left(\frac{1}{6}\right)^3 \\ &\quad = 420\left(\frac{1}{6}\right)^7 \end{aligned}$$

The multinomial is especially popular because of its use as a model of language. For a full example see the Federalist Paper Authorship example.

Continuous Joint

Random variables X and Y are Jointly Continuous if there exists a joint Probability Density Function (PDF) f such that:

$$P(a_1 < X \leq a_2, b_1 < Y \leq b_2) = \int_{a_1}^{a_2} \int_{b_1}^{b_2} f(X = x, Y = y)\, dy\ dx$$

Using the PDF we can compute marginal probability densities:

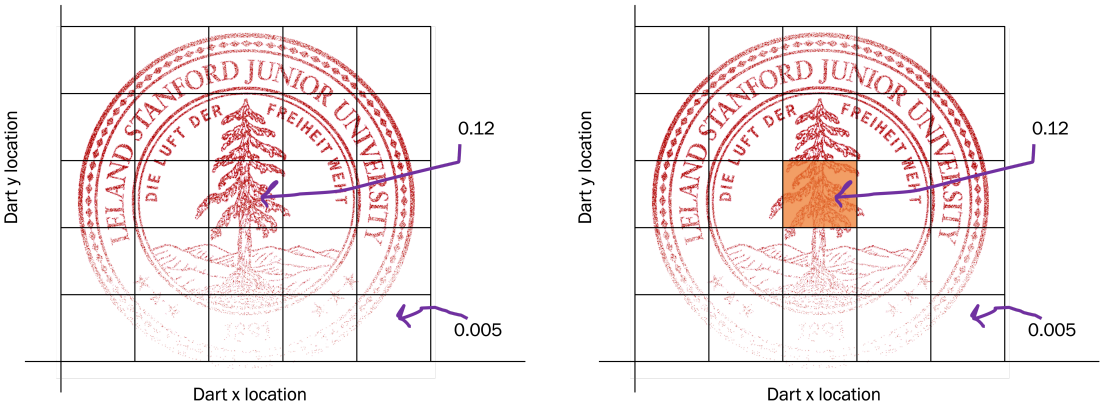
$$f(X = a) = \int_{-\infty}^{\infty} f(X = a, Y = y)\, dy$$
$$f(Y = b) = \int_{-\infty}^{\infty} f(X = x, Y = b)\, dx$$

Let $F(x, y)$ be the Cumulative Density Function (CDF):

$$P(a_1 < X \leq a_2, b_1 < Y \leq b_2) = F(a_2, b_2) - F(a_1, b_2) + F(a_1, b_1) - F(a_2, b_1)$$

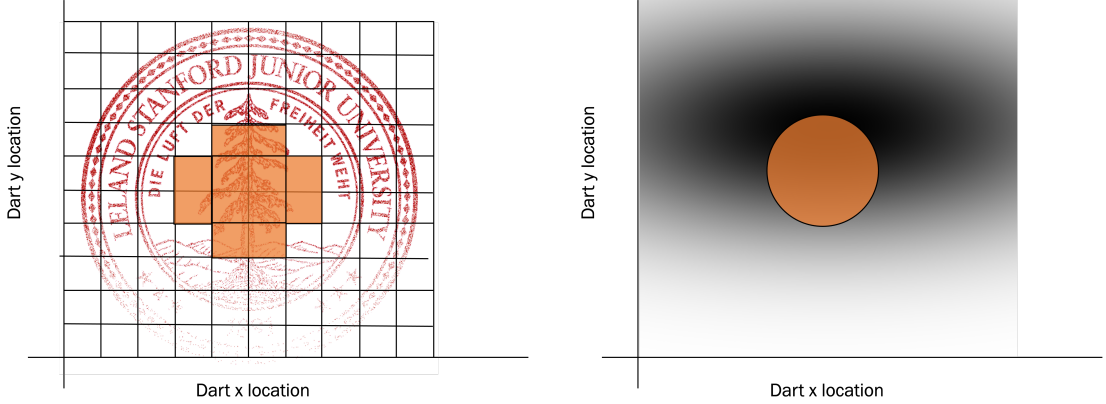
1. From Discrete Joint to Continuous Joint

Thinking about multiple continuous random variables jointly can be unintuitive at first blush. But we can turn to our helpful trick that we can use to understand continuous random variables: start with a discrete approximation. Consider the example of creating the CS109 seal. It was generated by throwing half a million darts at an image of the stanford logo (keeping all the pixels that get hit by at least one dart). The darts could hit any continuous location on the logo, and, the locations are not equally likely. Instead, the location a dart hits is goverened by a joint continuous distribution. In this case there are only two simultaneous random variables, the x location of the dart and the y location of the dart. Each random variable is continuous (it takes on real numbers). Thinking about the joint probability density function is easier by first considering a discretization. I am going to break the dart landing area into 25 discrete buckets:

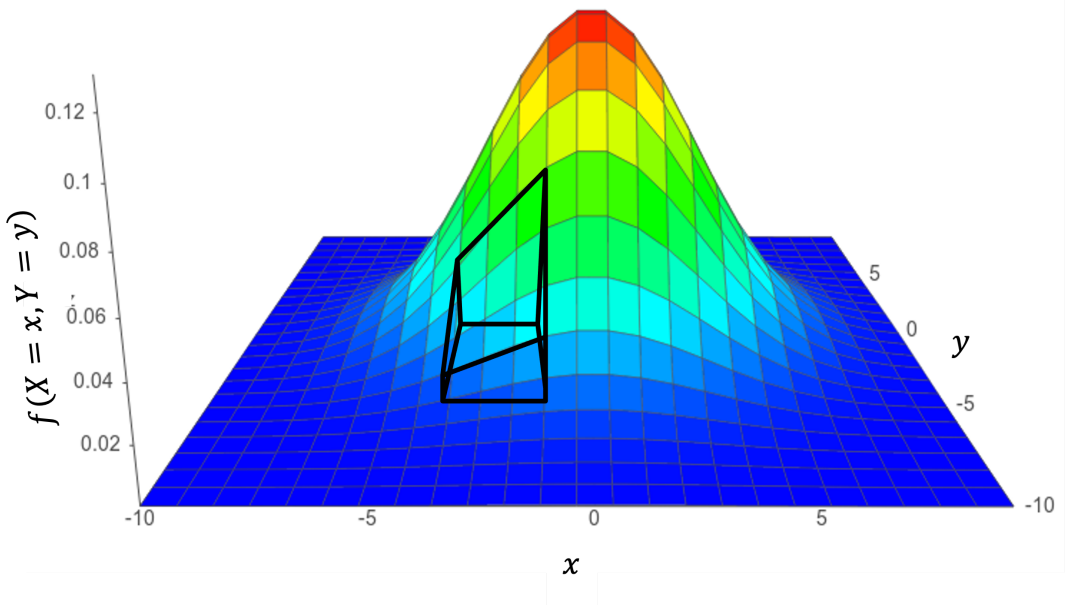


On the left is a visualization of the probability mass of this joint distribution, and on the right is a visualization of how we could answer the question: what is the probability that a dart hits within a certain distance of the center. For each bucket there is a single number, the probability that a dart will fall into that particular bucket (these probabilities are mutually exclusive and sum to 1).

Of course this discretization only *approximates* the joint probability distribution. In order to get a better approximation we could create more fine-grained discretizations. In the limit we can make our buckets infinitely small, and the value associated with each bucket becomes a second derivative of probability.



To represent the 2D probability density in a graph, we use the darkness of a value to represent the density (darker means more density). Another way to visualize this distribution is from an angle. This makes it easier to realize that this is a function with two inputs and one output. Below is an different visualization of the exact same density function:



Just like in the single random variable case, we are now represending our belief in the continuous random variables as *densities* rather than probabilities. Recall that a density represents a relative belief. If the density of $f(X = 1.1, Y = 0.9)$ is twice as high as the density that $f(X = 1.1, Y = 1.1)$ the function is expressing that it is twice as likely to find the particular combination of $X = 1.1$ and $Y = 0.9$.

2. Multivariate Gaussian

The density that is depicted in this example happens to be a particular of joint continuous distribution called Multivariate Gaussian. In fact it is a special case where all of the constituent variables are independent.

Def: Independent Multivariate Gaussian . An Independent Multivariate Gaussian can model a collection of continuous joint random variables $\vec{X} = (X_1 \dots X_n)$ as being a composition of independent normals with means $\vec{\mu} = (\mu_1 \dots \mu_n)$ and standard deviations $\vec{\sigma} = (\sigma_1 \dots \sigma_n)$. Notice how we now have variables in vectors (similar to a list in python). The notation for the multivariate uses vector notation:

$$\vec{X} \sim \vec{\mathrm{N}}(\vec{\mu}, \vec{\sigma})$$

The joint PDF is:

$$\begin{aligned} f(\vec{x}) &= \prod_{i=1}^{n} f(x_i) \\ &= \prod_{i=1}^{n} \frac{1}{\sigma_i \sqrt{2\pi}} e^{\frac{-(x-\mu_i)^2}{2\sigma_i^2}} \end{aligned}$$

And the joint CDF is

$$
\begin{aligned}
F(\vec{x}) &= \prod_{i=1}^{n} F(x_i) \\
&= \prod_{i=1}^{n} \Phi(\frac{x_i - \mu_i}{\sigma_i})
\end{aligned}
$$

Example: Gaussian Blur

In the same way that many single random variables are assumed to be gaussian, many joint random variables may be assumed to be Multivariate Gaussian. Consider this example of Gaussian Blur:

In image processing, a Gaussian blur is the result of blurring an image by a Gaussian function. It is a widely used effect in graphics software, typically to reduce image noise. A Gaussian blur works by convolving an image with a 2D independent multivariate gaussian (with means of 0 and equal valued standard deviations).

In order to use a Gaussian blur, you need to be able to compute the probability mass of that 2D gaussian in the space of pixels. Each pixel is given a weight equal to the probability that X and Y are both within the pixel bounds. The center pixel covers the area where $-0.5 \leq x \leq 0.5$ and $-0.5 \leq y \leq 0.5$. Let's do one step in computing the Gaussian function discretized over image space. What is the weight of the center pixel for gaussian blur with a multivariate gaussian which has means of 0 and standard deviation of 3?

Let $\vec{B}$ be the multivariate gaussian, $\vec{B} \sim \mathrm{N}(\vec{\mu} = [0, 0], \vec{\sigma} = [3, 3])$. Let's compute the CDF of this multivariate gaussian $F(x_1, x_2)$:

$$
\begin{aligned}
F(x_1, x_2) &= \prod_{i=1}^{n} \Phi(\frac{x_i - \mu_i}{\sigma_i}) \\
&= \Phi(\frac{x_1 - \mu_1}{\sigma_1}) \cdot \Phi(\frac{x_2 - \mu_2}{\sigma_2}) \\
&= \Phi(\frac{x_1}{3}) \cdot \Phi(\frac{x_2}{3})
\end{aligned}
$$

Now we are ready to calculate the weight of the center pixel:

$$
\begin{aligned}
&\mathrm{P}(-0.5 < X_1 \leq 0.5, -0.5 < X_2 \leq 0.5) \\
&\quad = F(0.5, 0.5) - F(-0.5, 0.5) + F(-0.5, -0.5) - F(0.5, -0.5) \\
&\quad = \Phi(\frac{0.5}{3}) \cdot \Phi(\frac{0.5}{3}) - \Phi(\frac{-0.5}{3}) \cdot \Phi(\frac{0.5}{3}) + \Phi(\frac{-0.5}{3}) \cdot \Phi(\frac{-0.5}{3}) - \Phi(\frac{0.5}{3}) \cdot \Phi(\frac{-0.5}{3}) \\
&\quad \approx 0.026
\end{aligned}
$$

How can this 2D gaussian blur the image? Wikipedia explains: "Since the Fourier transform of a Gaussian is another Gaussian, applying a Gaussian blur has the effect of reducing the image's high-frequency components; a Gaussian blur is a low pass filter" [2].

Inference

So far we have set the foundation for how we can represent probabilistic models with multiple random variables. These models are especially useful because they let us perform a task called "inference" where we update our belief about one random variable in the model, conditioned on new information about another. Inference in general is hard! In fact, it has been proven that in the worst case, the inference task, can be NP-Hard where n is the number of random variables [1].

First we are going to practice it with two random variables (in this section). Then, later in this unit we are going to talk about inference in the general case, with many random variables.

Earlier we looked at conditional probabilities for events. The first task in inference is to understand how to combine conditional probabilities and random variables. The equations for both the discrete and continuous case are intuitive extensions of our understanding of conditional probability:

1. The Discrete Conditional

The discrete case, where every random variable in your model is discrete, is a straightforward combination of what you know about conditional probability (which you learned in the context of events). Recall that every relational operator applied to a random variable defines an event. As such the rules for conditional probability directly apply: The conditional probability mass function (PMF) for the discrete case:

Let X and Y be discrete random variables.
Def: Conditional definition with discrete random variables.

$$\mathrm{P}(X = x|Y = y) = \frac{P(X = x, Y = y)}{P(Y = y)}$$

Def: Bayes' theorem with discrete random variables.

$$\mathrm{P}(X = x|Y = y) = \frac{P(Y = y|X = x)P(X = x)}{P(Y = y)}$$

In the presence of multiple random variables, it becomes increasingly useful to use shorthand! The above definition is identical to this notation where a lowercase symbol such as x is short hand for the event $X = x$:

$$\mathrm{P}(x|y) = \frac{P(x, y)}{P(y)}$$

The conditional definition works for any event and as such we can also write conditionals using cumulative density functions (CDFs) for the discrete case:

$$\begin{aligned}\mathrm{P}(X \leq a|Y = y) &= \frac{\mathrm{P}(X \leq a, Y = y)}{\mathrm{P}(Y = y)} \\ &= \frac{\sum_{x \leq a} \mathrm{P}(X = x, Y = y)}{\mathrm{P}(Y = y)}\end{aligned}$$

Here is a neat result: this last term can be rewritten, by a clever manipulation. We can make the sum extend over the whole fraction:

$$\begin{aligned}\mathrm{P}(X \leq a|Y = y) &= \frac{\sum_{x \leq a} \mathrm{P}(X = x, Y = y)}{\mathrm{P}(Y = y)} \\ &= \sum_{x \leq a} \frac{\mathrm{P}(X = x, Y = y)}{\mathrm{P}(Y = y)} \\ &= \sum_{x \leq a} \mathrm{P}(X = x|Y = y)\end{aligned}$$

In fact it becomes straight forward to translate the rules of probability (such as bayes theorem, law of total probability, etc) to the language of discrete random variables: we simply need to recall that every relational operator applied to a random variable defines an event.

2. Mixing Discrete and Continuous

What happens when we want to reason about *continuous* random variables using our rules of probability (such as Bayes theoreom, law of total probability, chain rule, etc)? There is a simple practical answer: the rules still apply, but we have to replace probability terminology with probability *density* functions. As a concrete example let's look at Bayes' Theorem with one continuous random variable.

Def: Bayes' Theorem with mixed discrete and continuous.

Let X be continuous random variable and let N be a discrete random variable. The conditional probabilities of X given N and N given X respectively are:

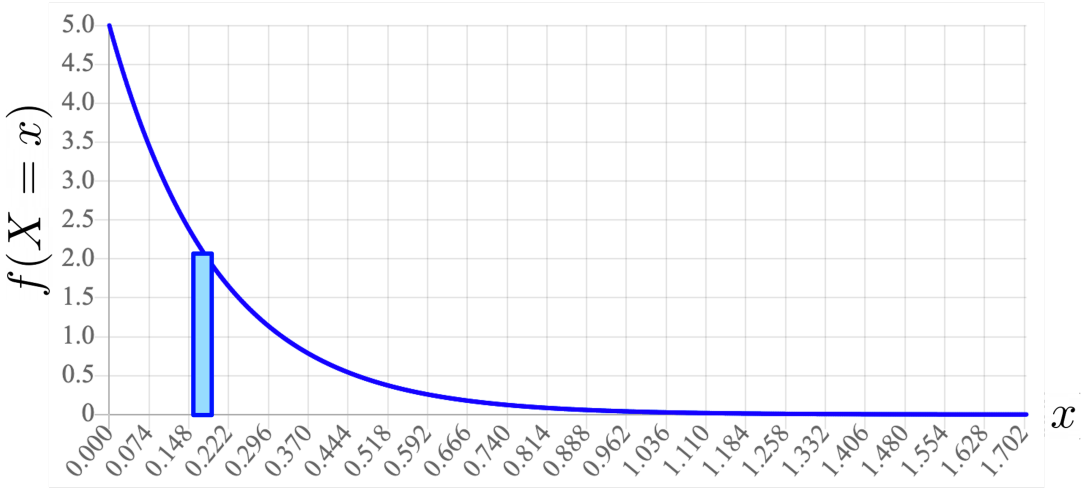
$$f(X = x|N = n) = \frac{\mathrm{P}(N = n|X = x)f(X = x)}{\mathrm{P}(N = n)}$$

$$\mathrm{P}(N = n|X = x) = \frac{f(X = x|N = n)\,\mathrm{P}(N = n)}{f(X = x)}$$

These equations might seem complicated since they mix probability densities and probabilities. Why should we believe that they are correct? First, observe that anytime the random variable on the left hand side of the conditional is continuous, we use a density, whenever it is discrete, we use a probability. This result can be derived by making the observation:

$$\mathrm{P}(X = x) = f(X = x) \cdot \epsilon_x$$

In the limit as $\epsilon_x \to 0$. In order to obtain a probability from a density function is to integrate under the function. If you wanted to approximate the probability that $X = x$ you could consider the area created by a rectangle which has height $f(X = x)$ and some very small width. As that width gets smaller, your answer becomes more accurate:



A value of ϵ_x is problematic if it is left in a formula. However, if we can get them to cancel, we can arrive at a working equation. This is the key insight used to derive the rules of probability in the context of one or more continuous random variables. Again, let X be continuous random variable and let N be a discrete random variable:

$$\begin{aligned}
\mathrm{P}(N = n|X = x) &= \frac{P(X = x|N = n)\,\mathrm{P}(N = n)}{P(X = x)} && \text{Bayes' Theorem} \\
&= \frac{f(X = x|N = n) \cdot \epsilon_x \cdot \mathrm{P}(N = n)}{f(X = x) \cdot \epsilon_x} && \mathrm{P}(X = x) = f(X = x) \cdot \epsilon_x \\
&= \frac{f(X = x|N = n) \cdot \mathrm{P}(N = n)}{f(X = x)} && \text{Cancel } \epsilon_x
\end{aligned}$$

This strategy applies beyond Bayes' Theorem. For example here is a version of the Law of Total Probability when X is continuous and N is discrete:

$$f(X=x) = \sum_{n \in N} f(X=x|N=n)\,\mathrm{P}(N=n)$$

3. Probability Rules with Continuous Random Variables

The strategy used in the above section can be used to derive the rules of probability in the presence of continuous random variables. The strategy also works when there are multiple continuous random variables. For example here is Bayes' Theorem with two continuous random variables.

Def: Bayes' Theorem with continuous random variables.

Let X and Y be continuous random variables.

$$f(X=x|Y=y) = \frac{f(X=x, Y=y)}{f(Y=y)}$$

4. Example: Inference with a Continuous Variable

Consider the following question:

Question: At birth, girl elephant weights are distributed as a Gaussian with mean 160kg, and standard deviation 7kg. At birth, boy elephant weights are distributed as a Gaussian with mean 165kg, and standard deviation of 3kg. All you know about a newborn elephant is that it is 163kg. What is the probability that it is a girl?

Answer: Let G be an indicator that the elephant is a girl. G is Bern(p = 0.5) Let X be the distribution of weight of the elephant.
$X|G=1$ is $N(\mu=160, \sigma^2=7^2)$
$X|G=0$ is $N(\mu=165, \sigma^2=3^2)$

$$\mathrm{P}(G=1|X=163) = \frac{f(X=163|G=1)\,\mathrm{P}(G=1)}{f(X=163)} \qquad \text{Bayes}$$

If we can solve this equation we will have our answer. What is $f(X=163|G=1)$? It is the probability density function of a gaussian X which has $\mu=160, \sigma^2=7^2$ at the point x is 163:

$$\begin{aligned} f(X=163|G=1) &= \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} && \text{PDF Gauss} \\ &= \frac{1}{7\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{163-160}{7}\right)^2} && \text{PDF } X \text{ at } 163 \end{aligned}$$

Next we note that $\mathrm{P}(G=0) = \mathrm{P}(G=1) = \frac{1}{2}$. Putting this all together, and using the law of total probability to compute the denominator we get:

$$
\begin{aligned}
&\mathrm{P}(G=1|X=163) \\
&= \frac{f(X=163|G=1)\,\mathrm{P}(G=1)}{f(X=163)} \\
&= \frac{f(X=163|G=1)\,\mathrm{P}(G=1)}{f(X=163|G=1)\,\mathrm{P}(G=1) + f(X=163|G=0)\,\mathrm{P}(G=0)} \\
&= \frac{\frac{1}{7\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{163-160}{7}\right)^2} \cdot \frac{1}{2}}{\frac{1}{7\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{163-160}{7}\right)^2} \cdot \frac{1}{2} + \frac{1}{3\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{163-165}{3}\right)^2} \cdot \frac{1}{2}} \\
&= \frac{\frac{1}{7} e^{-\frac{1}{2}\left(\frac{9}{49}\right)}}{\frac{1}{7} e^{-\frac{1}{2}\left(\frac{9}{49}\right)} + \frac{1}{3} e^{-\frac{1}{2}\left(\frac{4}{9}\right)^2}} \\
&\approx 0.328
\end{aligned}
$$

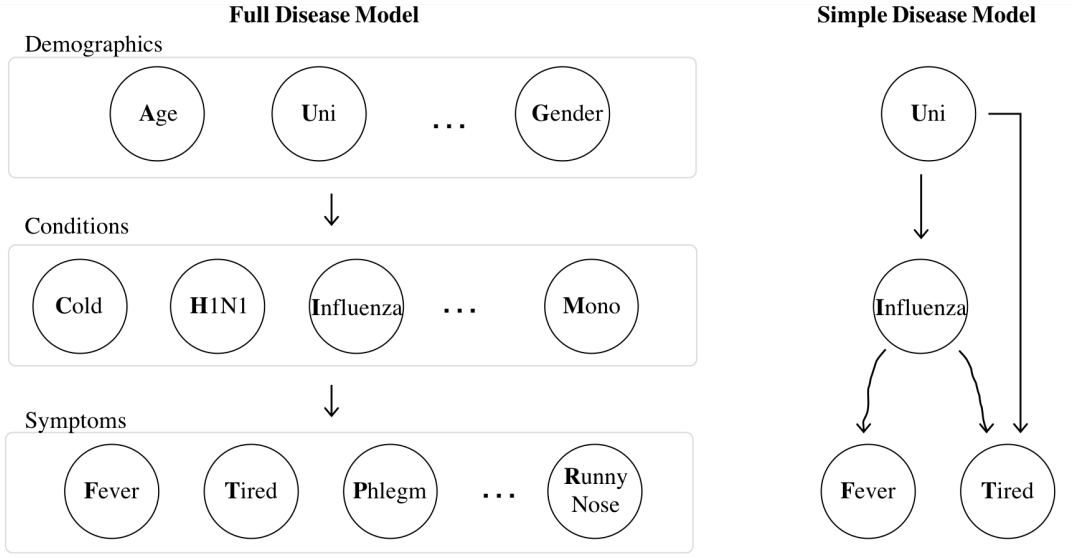
Bayesian Networks

At this point in the reader we have developed tools for analytically solving for probabilities. We can calculate the likelihood of random variables taking on values, even if they are interacting with other random variables (which we have called multi-variate models, or we say the random variables are jointly distributed). We have also started to study samples and sampling.

Consider the WebMd Symptom Checker. WebMD have built a probabilistic model with random variables which roughly fall under three categories: symptoms, risk factors and diseases. For any combination of observed symptoms and risk factors, they can calculate the probability of any disease. For example, they can calculate the probability that I have influenza given that I am a 21-year-old female who has a fever and who is tired: $P(I = 1|A = 21, G = 1, T = 1, F = 1)$. Or they could calculate the probability that I have a cold given that I am a 30-year-old with a runny nose: $P(C = 1|A = 30, R = 1)$. At first blush this might not seem difficult. But as we dig deeper we will realize just how hard it is. There are two challenges: (1) Modelling: sufficiently specifying the probabilistic model and (2) Inference: calculating any desired probability.

1. Bayesian Networks

Before we jump into how to solve probability (aka inference) questions, let's take a moment to go over how an expert doctor could specify the relationship between so many random variables. Ideally we could have our expert sit down and specify the entire "joint distribution" (see the first lecture on multi-variable models). She could do so either by writing a single equation that relates all the variables (which is as impossible as it sounds), or she could come up with a joint distribution table where she specifies the probability of any possible combination of assignments to variables. It turns out that is not feasible either. Why? Imagine there are $N = 100$ binary random variables in our WebMD model. Our expert doctor would have to specify a probability for each of the $2^N > 10^{30}$ combinations of assignments to those variables, which is approaching the number of atoms in the universe. Thankfully, there is a better way. We can simplify our task if we know the "generative" process that creates a joint assignment. Based on the generative process we can make a data structure known as a **Bayesian Network**. Here are two networks of random variables for diseases:



For diseases the flow of influence is directed. The states of "demographic" random variables influence whether someone has particular "conditions", which influence whether someone shows particular "symptoms". On the right is a simple model with only four random variables. Though this is a less interesting model it is easier to understand when first learning Bayesian Networks. Being in university (binary) influences whether or not someone has influenza (binary). Having influenza influences whether or not someone has a fever (binary) and the state of university and influenza influences whether or not someone feels tired (also binary).

In a Bayesian Network an arrow from random variable X to random variable Y articulates our assumption that X directly influences the likelihood of Y. We say that X is a *parent* of Y. To fully define the Bayesian network we must provide a way to compute the probability of each random variable (X_i) conditioned on knowing the value of all their parents: $P(X_i = k|\text{Parents of } X_i \text{ take on specified values})$. Here is a concrete example of what needs to be defined for the simple disease model. Recall that each of the random variables is binary:

$$P(\text{Uni} = 1) = 0.8$$
$$P(\text{Influenza} = 1|\text{Uni} = 1) = 0.2 \qquad P(\text{Fever} = 1|\text{Influenza} = 1) = 0.9$$
$$P(\text{Influenza} = 1|\text{Uni} = 0) = 0.1 \qquad P(\text{Fever} = 1|\text{Influenza} = 0) = 0.05$$
$$P(\text{Tired} = 1|\text{Uni} = 0, \text{Influenza} = 0) = 0.1 \qquad P(\text{Tired} = 1|\text{Uni} = 0, \text{Influenza} = 1) = 0.9$$
$$P(\text{Tired} = 1|\text{Uni} = 1, \text{Influenza} = 0) = 0.8 \qquad P(\text{Tired} = 1|\text{Uni} = 1, \text{Influenza} = 1) = 1.0$$

Let's put this in programming terms. All that we need to do in order to code up a Bayesian network is to define a function: **`getProbXi(i, k, parents)`** which returns the probability that X_i (the random var with index **`i`**) takes on the value **`k`** given a value for each of the parents of X_i encoded by the list **`parents`**: $P(X_i = x_i|\text{Values of parents of } X_i)$

Deeper understanding: The reason that a Bayes Net is so useful is that the "joint" probability can be expressed in exponentially less space as the product of the probabilities of each random variable conditioned on its parents! Without loss of generality, let X_i refer to the ith random variable (such that if X_i is a parent of X_j then $i < j$):

$$P(\text{Joint}) = P(X_1 = x_1, \ldots, X_n = x_n) = \prod_i P(X_i = x_i|\text{Values of parents of } X_i)$$

What assumptions are implicit in a Bayes' Net? Using the chain rule we can decompose the **exact** joint probability for n random variables. To make the following math easier to digest I am going to use x_i as shorthand for the event that $X_i = x_i$:

$$P(x_1, \ldots, x_n) = \prod_i P(x_i|x_{i-1}, \ldots, x_1)$$

By looking at the difference in the two equations, we can see that a Bayes' Net is assumping that

$$P(x_i|x_{i-1}, \ldots, x_1) = P(x_i|\text{Values of parents of } X_i)$$

This is a conditional independence statement. It is saying that once you know the value of the parents of a variable in your network, X_i, any further information about non-descendents will not change your belief in X_i. Formally we say that X_i is conditionally independent of its non-descendents, given its parents. What is a non-descendent again? In a graph, a descendent of X_i is anything which is in the subtree that starts at X_i. Everything else is a non-descendent. Non-descendents include the "ancestor" nodes of X_i as well as nodes which are totally unconnected to X_i. When designing Bayes' Nets you don't have to think about this assumption directly. It turns out to be a naturally good assumption if the arrows between your nodes follow a causal path.

2. Designing a Bayes Net

There are several steps to designing a Bayes Net.

1. Chose you random variables, and make them nodes.
2. Add edges, often based off your assumptions about which nodes directly cause which others.
3. Define $P(X_i = x_i|\text{Values of parents of } X_i)$ for all nodes.

As you might have guessed, we can do step (2) and (3) by hand, or, we can have computers try and perform those tasks based on data. The first task is called "structure learning" and the second is an instance of "machine learning." There are fully autonomous solutions to structure learning -- but they only work well if you have a massive amount of data. Alternatively people will often compute a statistic called correlation between all pairs of random variables to help in the art form of designing a Bayes' net.

In the next part of the reader we are going to talk about how we could learn $P(X_i = x_i|\text{Values of parents of } X_i)$ from data. For now let's start with the (reasonable) assumption that an expert can write down these functions in equation or as python: **`getProbXi`**.

3. Next Steps

Great! We have a feasible way to define a large network of random variables. First challenge complete. We haven't talked about continuous of multinomial random variables in Bayes Nets. None of the theory changes: the expert will just have to define `getProbXi` to handle more values of `k` than 0 or 1.

A Bayesian network is not very interesting to us unless we can use it to solve different conditional probability questions. How can we perform "inference" on a network as complex as a Bayesian network?

Independence in Variables

1. Discrete

Two discrete random variables X and Y are called independent if:

$$\mathrm{P}(X = x, Y = y) = \mathrm{P}(X = x)\,\mathrm{P}(Y = y) \text{ for all } x, y$$

Intuitively: knowing the value of X tells us nothing about the distribution of Y. If two variables are not independent, they are called dependent. This is a similar conceptually to independent events, but we are dealing with multiple *variables*. Make sure to keep your events and variables distinct.

2. Continuous

Two continuous random variables X and Y are called independent if:

$$\mathrm{P}(X \leq a, Y \leq b) = \mathrm{P}(X \leq a)\,\mathrm{P}(Y \leq b) \text{ for all } a, b$$

This can be stated equivalently using either the CDF or the PDF:

$$F_{X,Y}(a, b) = F_X(a)F_Y(b) \text{ for all } a, b$$
$$f(X = x, Y = y) = f(X = x)f(Y = y) \text{ for all } x, y$$

More generally, if you can factor the joint density function then your random variable are independent (or the joint probability function for discrete random variables):

$$f(X = x, Y = y) = h(x)g(y)$$
$$\mathrm{P}(X = x, Y = y) = h(x)g(y)$$

3. Example: Showing Independence

Let N be the # of requests to a web server/day and that $N \sim \mathrm{Poi}(\lambda)$. Each request comes from a human with probability = p or from a "bot" with probability = $(1–p)$. Define X to be the # of requests from humans/day and Y to be the # of requests from bots/day. Show that the number of requests from humans, X, is independent of the number of requests from bots, Y.

Since requests come in independently, the probability of X conditioned on knowing the number of requests is a Binomial. Specifically:

$$(X|N) \sim \mathrm{Bin}(N, p)$$
$$(Y|N) \sim \mathrm{Bin}(N, 1 - p)$$

To get started we need to first write an expression for the join probability of X and Y. To do so, we use the chain rule:

$$\mathrm{P}(X = x, Y = y) = \mathrm{P}(X = x, Y = y|N = x + y)\,\mathrm{P}(N = x + y)$$

We can calculate each term in this expression. The first term is the PMF of the binomial $X|N$ having x ``successes."" The second term is the probability that the Poisson N takes on the value $x + y$:

$$\mathrm{P}(X = x, Y = y|N = x + y) = \binom{x + y}{x} p^x (1 - p)^y$$
$$\mathrm{P}(N = x + y) = e^{-\lambda}\frac{\lambda^{x+y}}{(x + y)!}$$

Now we can put those together we have an expression for the joint:

$$\mathrm{P}(X = x, Y = y) = \binom{x + y}{x} p^x (1 - p)^y e^{-\lambda}\frac{\lambda^{x+y}}{(x + y)!}$$

At this point we have derived the joint distribution over X and Y. In order to show that these two are independent, we need to be able to factor the joint:

$$\begin{aligned}
&\mathrm{P}(X = x, Y = y) \\
&= \binom{x+y}{x} p^x (1-p)^y e^{-\lambda} \frac{\lambda^{x+y}}{(x+y)!} \\
&= \frac{(x+y)!}{x! \cdot y!} p^x (1-p)^y e^{-\lambda} \frac{\lambda^{x+y}}{(x+y)!} \\
&= \frac{1}{x! \cdot y!} p^x (1-p)^y e^{-\lambda} \lambda^{x+y} && \text{Cancel (x+y)!} \\
&= \frac{p^x \cdot \lambda^x}{x!} \cdot \frac{(1-p)^y \cdot \lambda^y}{y!} \cdot e^{-\lambda} && \text{Rearrange}
\end{aligned}$$

Because the joint can be factered into a term that only has x and a term that only has y, the random variables are independent.

4. Symmetry of Independence

Independence is symmetric. That means that if random variables X and Y are independent, X is independent of Y and Y is independent of X. This claim may seem meaningless but it can be very useful. Imagine a sequence of events $X_1, X_2, \ldots$. Let A_i be the event that X_i is a "record value" (eg it is larger than all previous values). Is A_{n+1} independent of A_n? It is easier to answer that A_n is independent of A_{n+1}. By symmetry of independence both claims must be true.

5. Expectation of Products

Lemma: Product of Expectation for Independent Random Variables:
If two random variables X and Y are independent, the expectation of their product is the product of the individual expectations.

$$\begin{aligned}
E[X \cdot Y] &= E[X] \cdot E[Y] && \text{if and only if } X \text{ and } Y \text{ are independent} \\
E[g(X)h(Y)] &= E[g(X)]E[h(Y)] && \text{where } g \text{ and } h \text{ are functions}
\end{aligned}$$

Note that this assumes that X and Y are independent. Contrast this to the sum version of this rule (expectation of sum of random variables, is the sum of individual expectations) which does **not** require the random variables to be independent.

Correlation

1. Covariance

Covariance is a quantitative measure of the extent to which the deviation of one variable from its mean matches the deviation of the other from its mean. It is a mathematical relationship that is defined as:

$$\text{Cov}(X, Y) = E[(X - E[X])(Y - E[Y])]$$

That is a little hard to wrap your mind around (but worth pushing on a bit). The outer expectation will be a weighted sum of the inner function evaluated at a particular (x, y) weighted by the probability of (x, y). If x and y are both above their respective means, or if x and y are both below their respective means, that term will be positive. If one is above its mean and the other is below, the term is negative. If the weighted sum of terms is positive, the two random variables will have a positive correlation. We can rewrite the above equation to get an equivalent equation:

$$\text{Cov}(X, Y) = E[XY] - E[Y]E[X]$$

Lemma: Correlation of Independent Random Variables:
If two random variables X and Y are independent, than their covariance must be 0.

$$\begin{aligned} \text{Cov}(X, Y) &= E[XY] - E[Y]E[X] && \text{Def of Cov} \\ &= E[X]E[Y] - E[Y]E[X] && \text{Lemma Product of Expectation} \\ &= 0 \end{aligned}$$

Note that the reverse claim is not true. Covariance of 0 does not prove independence.

Using this equation (and the product lemma) is it easy to see that if two random variables are independent their covariance is 0. The reverse is \emph{not} true in general.

2. Properties of Covariance

Say that X and Y are arbitrary random variables:

$$\begin{aligned} &\text{Cov}(X, Y) = \text{Cov}(Y, X) \\ &\text{Cov}(X, X) = E[X^2] - E[X]E[X] = \text{Var}(X) \\ &\text{Cov}(aX + b, Y) = a\text{Cov}(X, Y) \end{aligned}$$

Let $X = X_1 + X_2 + \cdots + X_n$ and let $Y = Y_1 + Y_2 + \cdots + Y_m$. The covariance of X and Y is:

$$\begin{aligned} \text{Cov}(X, Y) &= \sum_{i=1}^{n} \sum_{j=1}^{m} \text{Cov}(X_i, Y_j) \\ \text{Cov}(X, X) = \text{Var}(X) &= \sum_{i=1}^{n} \sum_{j=1}^{n} \text{Cov}(X_i, X_j) \end{aligned}$$

That last property gives us a third way to calculate variance. We can use it to, again, show how to get the variance of a Binomial.

3. Correlation

We left off last class talking about covariance. Covariance was interesting because it was a quantitative measurement of the relationship between two variables. Today we are going to extend that concept to correlation. Correlation between two random variables, $\rho(X, Y)$ is the covariance of the two variables normalized by the variance of each variable. This normalization cancels the units out:

$$\rho(X, Y) = \frac{\text{Cov}(X, Y)}{\sqrt{\text{Var}(X) Var(Y)}}$$

Correlation measure linearity between X and Y.

$$\begin{array}{ll} \rho(X,Y) = 1 & Y = aX + b \text{ where } a = \sigma_y/\sigma_x \\ \rho(X,Y) = -1 & Y = aX + b \text{ where } a = -\sigma_y/\sigma_x \\ \rho(X,Y) = 0 & \text{absence of linear relationship} \end{array}$$

If $\rho(X,Y) = 0$ we say that X and Y are ``uncorrelated."

When people use the term correlation, they are actually referring to a specific type of correlation called ``Pearson" correlation. It measures the degree to which there is a linear relationship between the two variables. An alternative measure is ``Spearman" correlation which has a formula almost identical to your regular correlation score, with the exception that the underlying random variables are first transformed into their rank. ``Spearman" correlation is outside the scope of CS109.

Computational Inference

A Bayesian Network gives us a reasonable way to specify the joint probability of a network of many random variables. Before we celebrate, realize that we still don't know how to use such a network to answer probability questions. There are many techniques for doing so. I am going to introduce you to one of the great ideas in probability for computer science: we can use sampling to solve inference questions on Bayesian networks. Sampling is frequently used in practice because it is relatively easy to understand and easy to implement.

1. Rejection Sampling

As a warmup consider what it would take to sample an assignment to each of the random variables in our Bayes net. Such a sample is often called a "joint sample" or a "particle" (as in a particle of sand). To sample a particle, simply sample a value for each random variable one at a time based on the value of the random variable's parents. This means that if X_i is a parent of X_j, you will have to sample a value for X_i before you sample a value for X_j.

Let's work through an example of sampling a "particle" for the Simple Disease Model in the Bayes Net section:
1. Sample from $P(\text{Uni} = 1)$: Bern(0.8). Sampled value for Uni is 1.
2. Sample from $P(\text{Influenza} = 1|\text{Uni} = 1)$: Bern$(0.2)$. Sampled value for Influenza is 0.
3. Sample from $P(\text{Fever} = 1|\text{Influenza} = 0)$: Bern$(0.05)$. Sampled value for Fever is 0.
4. Sample from $P(\text{Tired} = 1|\text{Uni} = 1, \text{Influenza} = 0)$: Bern$(0.8)$. Sampled value for Tired is 0.
Thus the sampled particle is: [Uni = 1, Influenza = 0, Fever = 0, Tired = 0]. If we were to run the process again we would get a new particle (with likelihood determined by the joint probability).

Now our strategy is simple: we are going to generate N samples where N is in the hundreds of thousands (if not millions). Then we can compute probability queries by counting. Let $N(\mathbf{X} = \mathbf{k})$ be notation for the number of particles where random variables \textbf{X} take on values \textbf{k}. Recall that the bold notation \textbf{X} means that \textbf{X} is a vector with one or more elements. By the "frequentist" definition of probability:

$$P(\mathbf{X} = \mathbf{k}) = \frac{N(\mathbf{X} = \mathbf{k})}{N}$$

Counting for the win! But what about conditional probabilities? Well using the definition of conditional probabilities, we can see it's still some pretty straightforward counting:

$$P(\mathbf{X} = \mathbf{a}|\mathbf{Y} = \mathbf{b}) = \frac{P(\mathbf{X} = \mathbf{a}, \mathbf{Y} = \mathbf{b})}{P(\mathbf{Y} = \mathbf{b})} = \frac{\frac{N(\mathbf{X}=\mathbf{a},\mathbf{Y}=\mathbf{b})}{N}}{\frac{N(\mathbf{Y}=\mathbf{b})}{N}} = \frac{N(\mathbf{X} = \mathbf{a}, \mathbf{Y} = \mathbf{b})}{N(\mathbf{Y} = \mathbf{b})}$$

Let's take a moment to recognize that this is straight-up fantastic. General inference based on analytic probability (math without samples) is hard even given a Bayesian network (if you don't believe me, try to calculate the probability of flu conditioning on one demographic and one symptom in the Full Disease Model). However if we generate enough samples we can calculate any conditional probability question by reducing our samples to the ones that are consistent with the condition ($\vec{Y} = \vec{b}$) and then counting how many of those are also consistent with the query ($\vec{X} = \vec{a}$). Here is the algorithm in pseudocode:

```
N = 10000
# "query" is the assignment to variables we want probabilities for
# condition" is the assignments to variables we will condition on
def getAnyProbability(query, condition):
  particles = generateManyJointSamples(N)
  condParticles = rejectNonConsistentSamples(particles, condition)
  K = countConsistentSamples(condParticles, query)
  return K / len(condParticles)
```

This algorithm is sometimes called "Rejection Sampling" because it works by generating many particles from the joint distribution and rejecting the ones that are not consistent with the set of assignments we are conditioning on. Of course this algorithm is an approximation, though with enough samples it often works out to be a very good approximation. However, in cases where the event we're conditioning on is rare enough that it doesn't occur after millions of samples are generated, our algorithm will not work. The last line of our code will result in a divide by 0 error. See the next section for solutions!

2. General Inference when Conditioning on Rare Events

Joint Sampling is a powerful technique that takes advantage of computational power. But it doesn't always work. In fact it doesn't work any time that the probability of the event we are conditioning is rare enough that we are unlikely to ever produce samples that exactly match the event. The simplest example is with continuous random variables. Consider the Simple Disease Model. Let's change Fever from being a binary variable to being a continuous variable. To do so the only thing we need to do is re-specify the likelihood of fever given assignments to its parents (influenza). Let's say that the likelihoods come from the normal PDF:

$$\text{if Influenza} = 0\text{, then Fever} \sim N(\mu = 98.3, \sigma = 0.7) \qquad \therefore f(\text{Fever} = x) = \frac{1}{\sqrt{2\pi \cdot 0.7}} e^{-\frac{(x-98.3)^2}{2 \cdot 0.7}}$$

$$\text{if Influenza} = 1\text{, then Fever} \sim N(\mu = 100.0, \sigma = 1.8) \qquad \therefore f(\text{Fever} = x) = \frac{1}{\sqrt{2\pi \cdot 1.8}} e^{-\frac{(x-100.0)^2}{2 \cdot 1.8}}$$

Drawing samples (aka particles) is still straightforward. We apply the same process until we get to the step where we sample a value for the Fever random variable (in the example from the previous section that was step 3). If we had sampled a 0 for influenza we draw a value for fever from the normal for healthy adults (which has $\mu = 98.3$). If we had sampled a 1 for influenza we draw a value for fever from the normal for adults with the flu (which has $\mu = 100.0$). The problem comes in the "rejection" stage of joint sampling.

When we sample values for fever we get numbers with infinite precision (eg 100.819238 etc). If we condition on someone having a fever equal to 101 we would reject every single particle. Why? No particle will have exactly a fever of 101.

There are several ways to deal with this problem. One especially easy solution is to be less strict when rejecting particles. We could round all fevers to whole numbers.

There is an algorithm called "Likelihood Weighting" which sometimes helps, but which we don't cover in CS109. Instead, in class we talked about a new algorithm called Markov Chain Monte Carlo (MCMC) that allowed us to sample from the "posterior" probability: the distribution of random variables after (post) us fixing variables in the conditioned event. The version of MCMC we talked about is called Gibbs Sampling. While I don't require that students in CS109 know how to implement Gibbs Sampling, I wanted everyone to know that it exists and that it isn't beyond your capabilities. If you need to use it, you can learn it given the knowledge you have now.

MCMC does require more math than Joint Sampling. For every random variable you will need to specify how to calculate the likelihood of assignments given the variable's: parents, children and parents of its children (a set of variables cozily called a "blanket"). Want to learn more? Take CS221 or CS228!

3. Thoughts

While there are slightly-more-powerful "general inference algorithms" that you will get to learn in the future, it is worth recognizing that at this point we have reached an important milestone in CS109. You can take very complicated probability models (encoded as Bayesian networks) and can answer general inference queries on them. To get there we worked through the concrete example of predicting disease. While the WebMd website is great for home users, similar probability models are being used in thousands of hospitals around the world. As you are reading this general inference is being used to improve health care (and sometimes even save lives) for real human beings. That's some probability for computer scientists that is worth learning. What if we don't have an expert? Could we learn those probabilities from data? Jump to part 5 to answer that question.

CS109 Logo

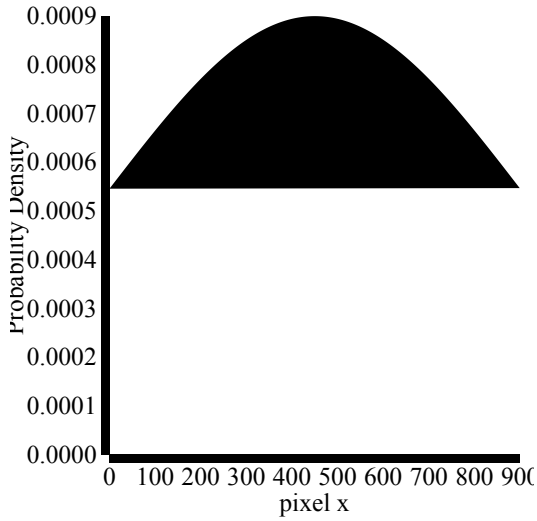
To generate the CS109 logo, we are going to throw half a million darts at a picture of the Stanford seal. We only keep the pixels that are hit by at least one dart. Each dart has it's x-pixel and y-pixel chosen at random from gaussian distributions. Let X be a random variable which represent the x-pixel, Y be a random variable which represents the y-pixel and S be a constant that equals the size of the logo (its width is equal to its height). $X \sim \mathcal{N}\left(\frac{S}{2}, \frac{S}{2}\right)$ and $Y \sim \mathcal{N}\left(\frac{S}{3}, \frac{S}{5}\right)$

Darts thrown: 0

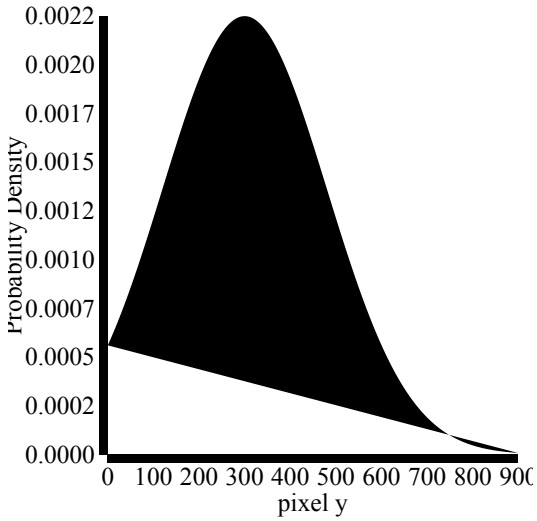
1. Dart Results

2. Dart Probability Density

3. X Distribution



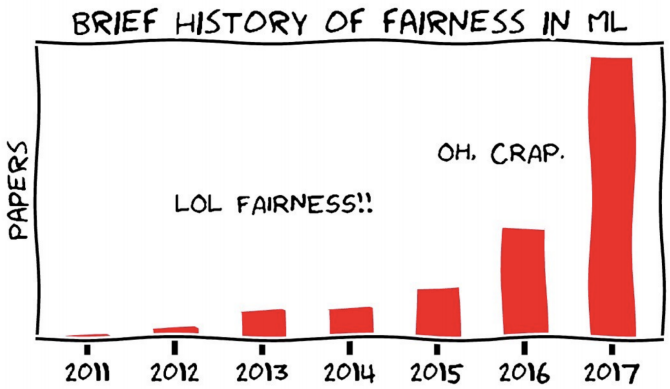
4. Y Distribution



Fairness in Artificial Intelligence

Artificial Intelligence often gives the impression that it is objective and "fair". However, algorithms are made by humans and trained by data which may be biased. There are several examples of AI algorithms, deployed, have been shown to make decisions that were biased based on gender, race or other protected demographics -- even when there is no intention for it.

These examples have also led to a necessary research into a growing field of algorithmic fairness. How can we demonstrate, or proove, that an algorithm is behaving in a way that we think is appropriate? What is fair? Clearly these are complex questions and are deserving of a complete conversation. This example is simple for the purpose of giving an introduction to the topic.



ML stands for Machine Learning. Solon Barocas and Moritz Hardt, "Fairness in Machine Learning", NeurIPS 2017

1. What is Fairness?

An artificial intelligence algorithm is going to be used to make a binary prediction (G for guess) for whether a person will repay a loan. The question has come up: is the algorithm "fair" with respect to a binary protected demographic (D for demographic)? To answer this question we are going to analyze predictions the algorithm made on historical data. We are then going to compare the predictions to the true outcome (T for truth). Consider the following joint probability table from the history of the algorithm's predictions:

	$D = 0$	
	$G = 0$	$G = 1$
$T = 0$	0.21	0.32
$T = 1$	0.07	0.28

	$D = 1$	
	$G = 0$	$G = 1$
$T = 0$	0.01	0.01
$T = 1$	0.02	0.08

Recall that cell $D = i, G = j, T = k$ contains the probability $\mathrm{P}(D = i, G = j, T = k)$. A joint probability table gives the probability of all combination of events. Recall that since each cell is mutually exclusive, the $\sum_i \sum_j \sum_k \mathrm{P}(D = i, G = j, T = k) = 1$. Note that this assumption of mutual exclusion could be problematic for demographic variables (some people are mixed ethnicity, etc) which gives you a hint that we are just scratching the surface in our conversation about fairness. Lets use this joint probability to learn about some of the common definitions of fairness.

Practice with joint marginalization

What is $\mathrm{P}(D = 0)$? What is $\mathrm{P}(D = 1)$?

Probabilities with assignments to a subset of the random variables in the joint distribution can be calculated by a process called *marginalization*: sum the probability from all cells where that assignment is true.

$$\begin{aligned}\mathrm{P}(D=1) &= \sum_{j\in\{0,1\}}\sum_{k\in\{0,1\}} \mathrm{P}(D=1, G=j, T=k) \\ &= 0.01 + 0.01 + 0.02 + 0.08 = 0.12\end{aligned}$$

$$\begin{aligned}\mathrm{P}(D=0) &= \sum_{j\in\{0,1\}}\sum_{k\in\{0,1\}} \mathrm{P}(D=0, G=j, T=k) \\ &= 0.21 + 0.32 + 0.07 + 0.28 = 0.88\end{aligned}$$

Note that $\mathrm{P}(D=0) + \mathrm{P}(D=1) = 1$. That implies that the demographics are mututally exclusive.

Fairness definition #1: Parity
An algorithm satisfies "parity" if the probability that the algorithm makes a positive prediction ($G = 1$) is the same regardless of begin conditioned on demographic variable.

Does this algorithm satisfy parity?

$$\begin{aligned}P(G=1|D=1) &= \frac{P(G=1, D=1)}{P(D=1)} && \text{Cond. Prob.} \\ &= \frac{P(G=1, D=1, T=0) + P(G=1, D=1, T=1)}{P(D=1)} && \text{Prob or} \\ &= \frac{0.01 + 0.08}{0.12} = 0.75 && \text{From joint} \\ P(G=1|D=0) &= \frac{P(G=1, D=0)}{P(D=0)} && \text{Cond. Prob.} \\ &= \frac{P(G=1, D=0, T=0) + P(G=1, D=0, T=1)}{P(D=0)} && \text{Prob or} \\ &= \frac{0.32 + 0.28}{0.88} \approx 0.68 && \text{From joint}\end{aligned}$$

No. Since $P(G=1|D=1) \neq P(G=1|D=0)$ this algorithm does not satisfy parity. It is more likely to guess 1 when the demographic indicator is 1.

Fairness definition #2: Calibration
An algorithm satisfies "calibration" if the probability that the algorithm is correct ($G = T$) is the same regardless of demographics.

Does this algorithm satisfy calibration?

The algorithm satisfies calibration if $P(G=T|D=0) = P(G=T|D=1)$

$$\begin{aligned}P(G=T|D=0) &= P(G=1, T=1|D=0) + P(G=0, T=0|D=0) \\ &= \frac{0.28 + 0.21}{0.88} \approx 0.56 \\ P(G=T|D=1) &= P(G=1, T=1|D=1) + P(G=0, T=0|D=1) \\ &= \frac{0.08 + 0.01}{0.12} = 0.75\end{aligned}$$

No: $P(G=T|D=0) \neq P(G=T|D=1)$

Fairness definition #3: Equality of Odds
An algorithm satisfies "equality of odds" if the probability that the algorithm *predicts a positive outcome* ($G = 1$) is the same regardless of demographics *given* that the outcome will occur ($T = 1$).

Does this algorithm satisfy equality of odds?

The algorithm satisfies equality of odds if $P(G=1|D=0, T=1) = P(G=1|D=1, T=1)$

$$\begin{aligned}P(G=1|D=1, T=1) &= \frac{P(G=1, D=1, T=1)}{P(D=1, T=1)} \\ &= \frac{0.08}{0.08 + 0.02} = 0.8 \\ P(G=1|D=0, T=1) &= \frac{P(G=1, D=0, T=1)}{P(D=0, T=1)} \\ &= \frac{0.28}{0.28 + 0.07} = 0.8\end{aligned}$$

Yes: $P(G=1|D=0, T=1) = P(G=1|D=1, T=1)$

Which of these definitions seems right to you? One can prove that you cant satisfy all of the above simultaneously. For a deeper treatment of the subject, here is a useful summary of the latest research Pessach et al. Algorithmic Fairness.

2. Gender Shades

In 2018, Joy Buolamwini and Timnit Gebru had a breakthrough result called "gender shades" published in the first conference on Fairness, Accountability and Transparency in ML [1]. They showed that facial recognition algorithms, which had been deployed to be used by Facebook, IBM and Microsoft, were substantially better at making predicitons (in this case classifying gender) when looking at lighter skinned men than darker skinned women. Their work exposed several shortcomings in production AI: biased datasets, optimizing for average accuracy (which means that the majority demographic gets most weight) lack of awarness of intersectionality, and more. Let's take a look at some of their results.

Figure by Joy Buolamwini and Timnit Gebru. Facial recognition algorithms perform very differently depending on who they are looking at. [1]

Timnit and Joy looked at three classifiers trained to predict gender, and computed several statistics. Lets take a look at one statistic, accuracy, for one of the facial recognition classifiers, IBMs:

	Women	**Men**	**Darker**	**Lighter**
Accuracy	79.7	94.4	77.6	96.8

Using the language of fairness, accuracy measures $\mathrm{P}(G = T)$. The cell in the table above under "Women" says the accuracy when looking at photos of women $\mathrm{P}(G = T | D = \text{Women})$. It is easy to show that these production level systems are terribly "uncalibrated":

$$\mathrm{P}(G = T | D = \text{Woman}) \neq \mathrm{P}(G = T | D = \text{Man})$$

$$\mathrm{P}(G = T | D = \text{Lighter}) \neq \mathrm{P}(G = T | D = \text{Darker})$$

Why should we care about callibration and not the other definitions of fairness? In the case the classifier was making a prediction of gender where a positive prediction (say predicting women) doesn't have directly associated reward as in our above example, where we were predicting if someone should receive a loan. As such the most salient idea is: is the algorithm just as accurate for different genders (callibration)?

The lack of callibration between men/women and lighter/darker skined photos is an issue. What Joy and Timnit showed next was that the problem becomes even worse when you look at intersectional demographics.

	Darker Men	**Darker Women**	**Lighter Men**	**Lighter Women**
Accuracy	88.0	65.3	99.7	92.9

If the algorithms were "fair" according to the callibration you would expect the accuracy to be the same regardness of demographics. Instead there is alomst a 34.2 percentage point difference! $\mathrm{P}(G = T | D = \text{Darker Woman})$ = 65.3 compared to $\mathrm{P}(G = T | D = \text{Ligher Man}) = 99.7$

[1] Buolamwini, Gebru. Gender Shades. 2018

3. Ways Forward?

Wadsworth et al. Achieving Fairness through Adversarial Learning

Federalist Paper Authorship

Let's write a program to decide whether or not James Madison or Alexander Hamilton wrote Fedaralist Paper 49. Both men have claimed to be have written it, and hence the authorship is in dispute. First we used historical essays to estimate p_i, the probability that Hamilton generates the word i (independent of all previous and future choices or words). Similarly we estimated q_i, the probability that Madison generates the word i. For each word i we observe the number of times that word occurs in Fedaralist Paper 49 (we call that count c_i). We assume that, given no evidence, the paper is equally likely to be written by Madison or Hamilton.

Define three events: H is the event that Hamilton wrote the paper, M is the event that Madison wrote the paper, and D is the event that a paper has the collection of words observed in Fedaralist Paper 49. We would like to know whether $P(H|D)$ is larger than $P(M|D)$. This is equivalent to trying to decide if $P(H|D)/P(M|D)$ is larger than 1.

The event $D|H$ is a multinomial parameterized by the values p. The event $D|M$ is also a multinomial, this time parameterized by the values q.

Using Bayes Rule we can simplify the desired probability.

$$\frac{P(H|D)}{P(M|D)} = \frac{\frac{P(D|H)P(H)}{P(D)}}{\frac{P(D|M)P(M)}{P(D)}} = \frac{P(D|H)P(H)}{P(D|M)P(M)} = \frac{P(D|H)}{P(D|M)}$$
$$= \frac{\binom{n}{c_1,c_2,\ldots,c_m}\prod_i p_i^{c_i}}{\binom{n}{c_1,c_2,\ldots,c_m}\prod_i q_i^{c_i}} = \frac{\prod_i p_i^{c_i}}{\prod_i q_i^{c_i}}$$

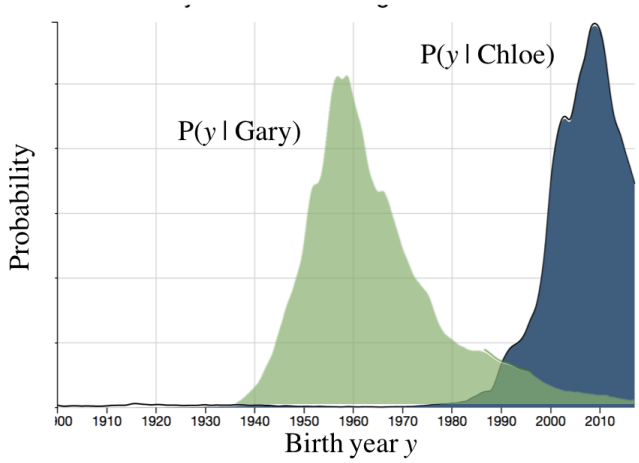
This seems great! We have our desired probability statement expressed in terms of a product of values we have already estimated. However, when we plug this into a computer, both the numerator and denominator come out to be zero. The product of many numbers close to zero is too hard for a computer to represent. To fix this problem, we use a standard trick in computational probability: we apply a log to both sides and apply some basic rules of logs.

$$\begin{aligned}\log\Big(\frac{P(H|D)}{P(M|D)}\Big) &= \log\Big(\frac{\prod_i p_i^{c_i}}{\prod_i q_i^{c_i}}\Big) \\ &= \log(\prod_i p_i^{c_i}) - \log(\prod_i q_i^{c_i}) \\ &= \sum_i \log(p_i^{c_i}) - \sum_i \log(q_i^{c_i}) \\ &= \sum_i c_i \log(p_i) - \sum_i c_i \log(q_i)\end{aligned}$$

This expression is ``numerically stable" and my computer returned that the answer was a negative number. We can use exponentiation to solve for $P(H|D)/P(M|D)$. Since the exponent of a negative number is a number smaller than 1, this implies that $P(H|D)/P(M|D)$ is smaller than 1. As a result, we conclude that Madison was more likely to have written Federalist Paper 49. That is the standing assuption currently made by historians!

Name to Age

Because of shifting patterns in name popularity, a person's age is a hint as to their age. The United States publishes a data which contains counts of how many US residents were born with a given name in a given year, based off Social Security applications. We can use inference to compute the reverse probability distribution: an updated belief in a person's age, given their name. As a reminder, if I know the year someone was born, I can calculate their age within one year.



The US Social Security applications data provides you with a function: `count(year, name)` which returns the number of US citizens, born in a given year with a given name. You also have access to a list `names` which has each name ever given in the US and `years` which has all the years. This function is implicitly giving us the joint probability over names and birth year. The probability of a joint assignment to name and birth year can be estimated as the count of people with that name, born on that year, over the total number of people in the dataset. Let B be the year someone is born, and let N be their name:

$$P(B = b, N = n) \approx \frac{\text{count}(b, n)}{\sum_{i \in \text{names}} \sum_{j \in \text{years}} \text{count}(i, j)}$$

The question we would really like to answer is: what is your belief that a resident was born in 1950, given that their name is Gary? We can get started by applying the definition of conditional probability for random variables:

$$\text{P}(B = 1950 | N = \text{Gary}) = \frac{\text{P}(N = \text{Gary}, B = 1950)}{\text{P}(N = \text{Gary})}$$

But this leaves one term to compute: $\text{P}(N = \text{Gary})$ which we can compute using marginalization:

$$\begin{aligned} \text{P}(N = \text{Gary}) &= \sum_{y \in \text{years}} P(B = y, N = \text{Gary}) \\ &\approx \frac{\sum_{y \in \text{years}} \text{count}(y, \text{Gary})}{\sum_{i \in \text{names}} \sum_{j \in \text{years}} \text{count}(i, j)} \end{aligned}$$

Putting this all together we have:

$$\begin{aligned} \text{P}(B = 1950 | N = \text{Gary}) &= \frac{\text{P}(N = \text{Gary}, B = 1950)}{\text{P}(N = \text{Gary})} \\ &\approx \frac{\left(\frac{\text{count}(1950, \text{Gary})}{\sum_{i \in \text{names}} \sum_{j \in \text{years}} \text{count}(i, j)} \right)}{\left(\frac{\sum_{y \in \text{years}} \text{count}(y, \text{Gary})}{\sum_{i \in \text{names}} \sum_{j \in \text{years}} \text{count}(i, j)} \right)} \\ &\approx \frac{\text{count}(1950, \text{Gary})}{\sum_{y \in \text{years}} \text{count}(y, \text{Gary})} \end{aligned}$$

More generally, for any name, we can compute the conditional probability mass function over birth year B:

$$\mathrm{P}(B = b | N = n) \approx \frac{\mathrm{count}(b, n)}{\sum_{y \in \mathrm{years}} \mathrm{count}(y, n)}$$

From Birth Year to Age

Of course, if B is the birth year of a person, their age, A is approximately the current year minus B. This could be off by one if someone has a birth day later in the year, but we will ignore this small deviation for now. So for example, if we think that a person was born in 1988, since the current year is 2022 then their age is 2022 - 1988 = 34

Assumptions

This problem makes many assumptions which are worth highlighting. In fact, any time we make generalizations (especially about demographics) based on sparse information we should tread lightly. Here are the assumptions that I can think of:

1. This data only is accurate for names of people in the US. The probability of age given names could be very different in other countries.
2. The US census is not perfect. It does not capture all people who are resident in the US, and there are demographics which are underrepresented. This will also skew our results.

Demo

Query Name: Katherine

Records with name:

This demo is based on real data from US Social Security applications between 1914 and 2014. Thank you to https://www.kaggle.com/kaggle/us-baby-names for compiling the data. Download Data.

Names that Give Away Your Age

Some names have certain years where they were exceptionally popular. These names provide quite a lot of information about birth year. Let's look at some of the names with the highest max probability.

Medium Popularity (>10,000 people with the name)

Name	Age with max prob	Prob of most likely age
Katina	49	0.245
Marquita	38	0.233
Ashanti	19	0.250
Miley	13	0.250
Aria	7	0.247

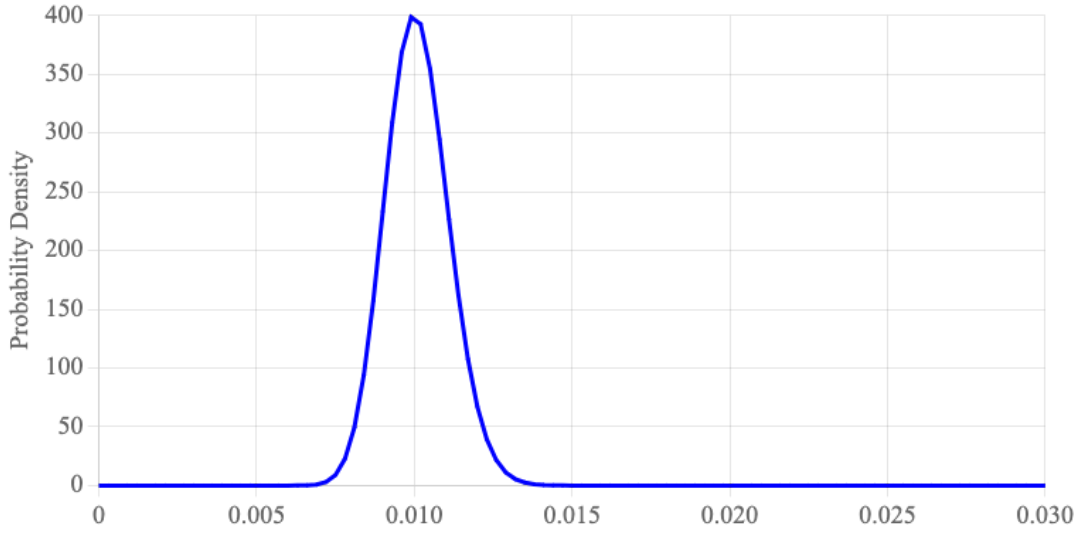
High Popularity (>100,000 people with the name)

Name	Age with max prob	Prob of most likely age
Debbie	62	0.104
Whitney	35	0.098
Chelsea	29	0.103
Aidan	18	0.098
Addison	14	0.112

A search for "Katina 1972" brought up this interesting article about a baby named Katina in a 1972 CBS Soap Opera. Marquita's popularity was likely from a 1983 toothpase add. Ashanti Douglas and Miley Cirus were popular singers in 2002 and 2008 respectively.

Futher Reading

Some names don't seem to have enough data to make good probability estimates. Can we quantify our uncertainty in such probability estimates? For example, if a name has only 10,000 entries in the database, of which only 100 were born in the year 1950, how confident are we that the true probability for 1950 is $\frac{100}{10000} = 0.01$? One way to express our uncertainty would be through a Beta Distribution. In this scenario we could represent our belief in the probability for 1950 as $X \sim \text{Beta}(a = 101, b = 9901)$ reflecting that we have seen 100 people born in 1950 and 9900 people who were not. We can plot that belief, zoomed into the range [0, 0.03]:



We can now ask questions such as, what is the probability that X is within 0.002 of 0.01?

$$\begin{aligned} &P(0.008 < X < 0.012) \\ &\quad = P(X < 0.012) - P(X < 0.008) \\ &\quad = F_X(0.012) - F_X(0.008) \\ &\quad = 0.966 - 0.013 \\ &\quad = 0.953 \end{aligned}$$

Semantically this leads to the claim that, after observing 100 births with a name in 1950, out of 10,000 births with that name over the whole dataset, there is a 95% chance that the probability of someone being born in 1950 is 0.010 ± 0.002.

Bridge Card Game

Stub: This section is not complete. Parts might not be fully written

Bridge is one of the most popular collaborative card games. It is played with four players in two teams. A few interesting probability problems come up in this game. You do not need to know the rules of bridge to follow this example.

1. Distribution of Suit Splits

When playing the game there are many times when one player will know *exactly* how many cards there are of a certain suit between their two opponents hands (call the opponents A and B). However, the player won't know the "split": how many of that particular suit are in opponent A's hand and how many cards of that suit are in opponent B's hand.

Both opponents have equal sized hands with k cards left. Across the two hands there are a known number of cards of a particular suit (eg spades) n, and you want to know how many are in one hand and how many are in the other. A split is represented as a tuple. For example $(0, 5)$ would mean 0 cards of the suit in opponent A's hands and 5 in opponent B's. Feel free to chose specific values for k and n:

k, the number of cards in each player's hand: 13

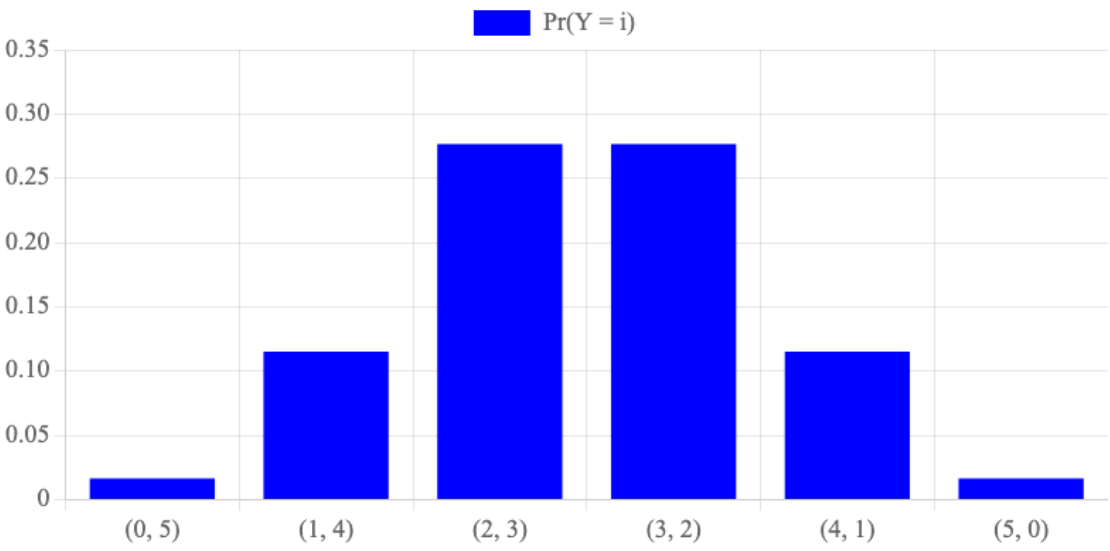
n, the number of cards of particular suit among the two hands: 5

A few notes: If there are k cards in each of the 2 hands there are $2k$ cards total. At the start of a game of bridge $k = 13$. It must be the case that $n \leq 2k$ because you can't have more cards of the suit left than number of cards! If there are n of a suit, then there are $2k - n$ of other suits. This problem assumes that the cards are properly shuffled.

Probability of different splits of the suit:
Let Y be a random variable representing the number of the suit in opponent A's hand. We can calculate the probability that Y equals different values i by counting equally likely outcomes.

$$\mathrm{P}(Y = i) = \frac{\binom{n}{i} \cdot \binom{2 \cdot k - n}{k - i}}{\binom{2 \cdot k}{k}}$$

Can you figure out how we came up with that formula? It uses equally likely outcomes where each element in the sample set is a chosen set of k cards to be dealt to one player (out of $2k$ cards which go to both). For k =13 and n =5 here is the PMF over splits:

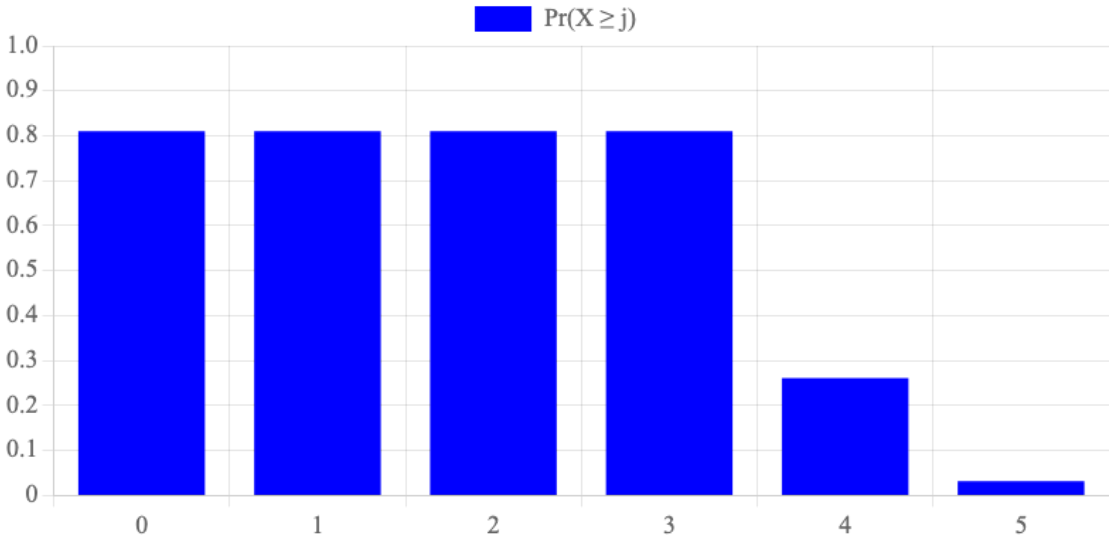


If we want to think about the probability of a given split, it is sufficient to chose one hand (call it "hand one") and think about the probability of the number of the given suit in that hand. Though there are two hands, if I tell you how many of a suit are in one hand, you can automatically figure out how many of the suit are in the other hand: recall that the number of the suit sums to n.

Probability that either hand has at least j cards of suit

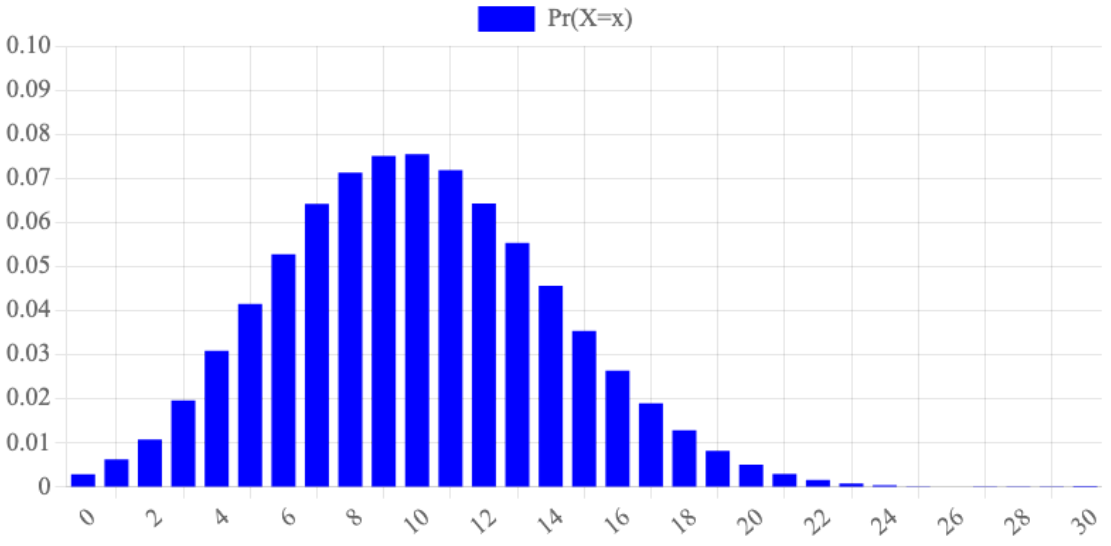
Let X be a random variable representing the highest number of cards of the suit in *either* hand. We can calculate the probability by using probability of or.

$$\mathrm{P}(X \geq j) = 1 - \sum_{i=n-j+1}^{j-1} \mathrm{P}(Y = i)$$



2. Distribution of Hand Strength

The way folks play bridge is that they make a calculation about their "hand strength" and then make decisions based off that number. The strength of your hand is a number which is equal to 4 times the number of "aces", 3 times the number of "kings", 2 times the number of "queens" and 1 times the number of "jacks" in your hand. No other cards contribute to your hand strength. Lets consider your hand strength to be a random variable and compute its distribution. It seems complex to compute by hand -- but perhaps we could run a simulation? Here we simulate a million deals of bridge hands, calculate the hand strengths, and use that to approximate the the distribution of hand strengths:



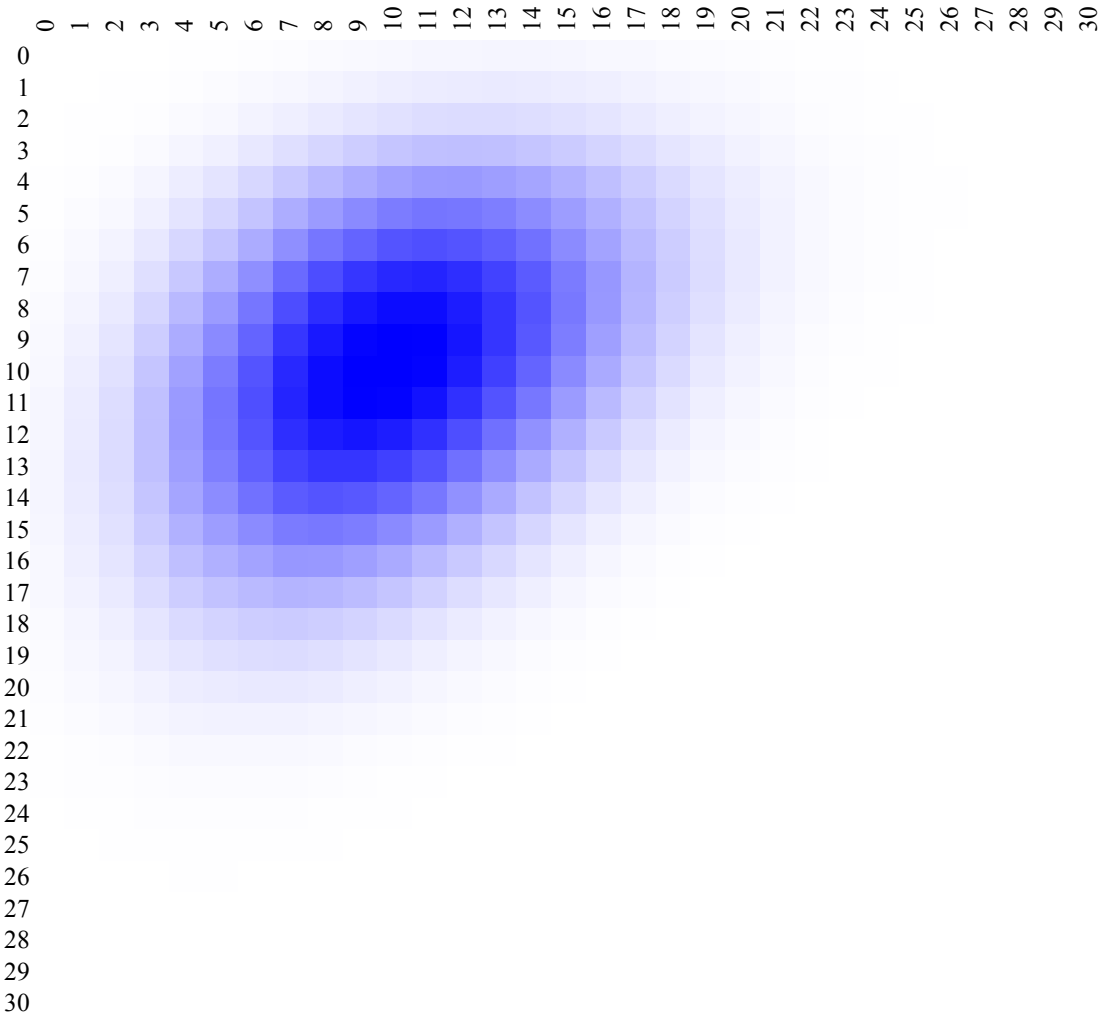
You might notice that at first blush this looks a lot like a poisson with rate $\lambda = 10$. First, lets consider why the rate might be 10. Let X_i be the points of a given card i. Since each card value is equally likely $\mathrm{P}(X_i = x) = \frac{1}{13}$. The expectation of points for each card is $\mathrm{E}[X] = \sum_x x \cdot \mathrm{P}(X_i = x) = (1 + 2 + 3 + 4)\frac{1}{13}$. Let H be the value of a hand. The value of a hand is the sum of the value of each card:

$$\begin{aligned}\mathrm{E}[H] &= \sum_{i \in \{1 \ldots 13\}} \mathrm{E}[X_i] \\ &= 13 \cdot \mathrm{E}[X_i] \\ &= 13 \cdot (1 + 2 + 3 + 4)\frac{1}{13} = 10\end{aligned}$$

Saying that H is approximately $\sim \mathrm{Poi}(\lambda = 10)$ is an interesting claim. It suggests that points in a hand come at a constant rate, and that the next point in your hand is independent of when you got your last point. Of course this second part of the assumption is mildly violated. There are a fixed set of cards so getting one card changes the probabilities of others. For this reason the poisson is a close, but not perfect approximation.

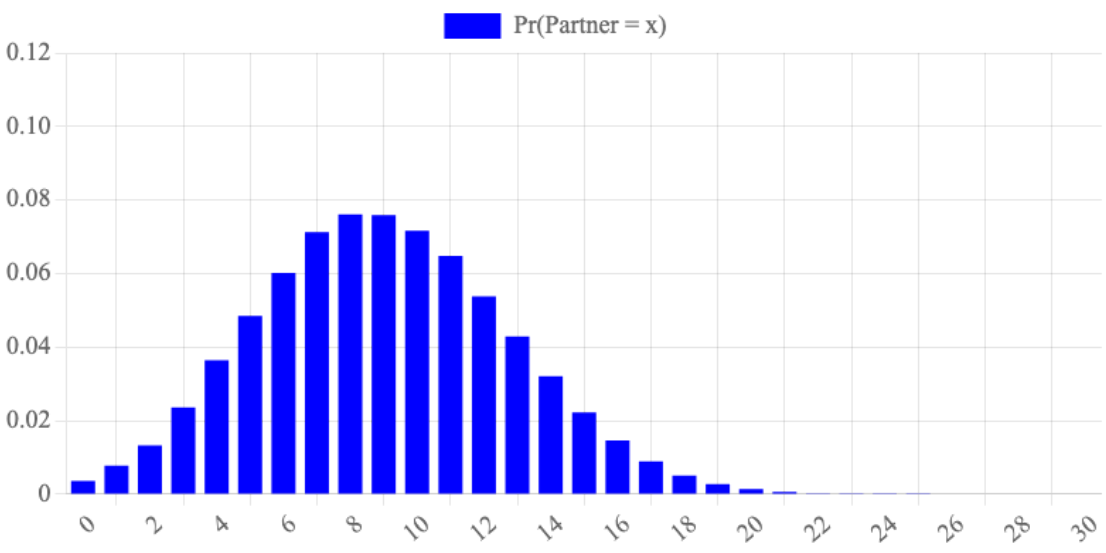
3. Joint distribution of hand strength among two hands

In most card games it doesn't just matter how strong your hand is, but the relative strength of your hand and another hand. In bridge, you play with a partner. We know that the two hands are not independent of each other. If I tell you that your partner has a strong hand, that means there are fewer "high value" cards that can be in your hand, and as such by belief in your strength has changed. If you think about each player's hand strength as a random variable, we care about the joint distribution of hand strength.



Finally lets consider the conditional distribution of your partners points given your points:

Your points: 13

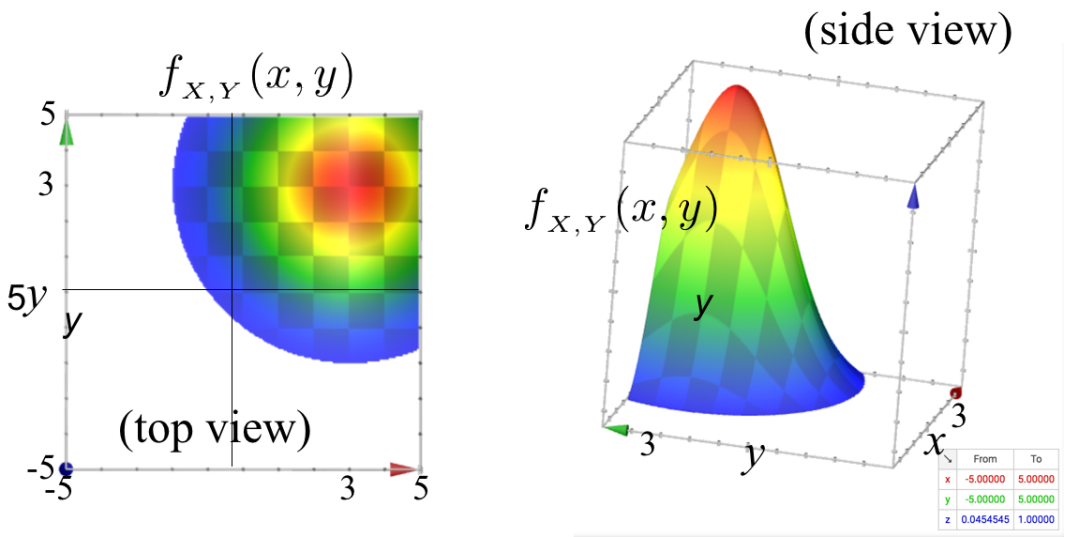

Pr(Partner = x)
0.12
0.10
0.08
0.06
0.04
0.02
0
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30

Tracking in 2D

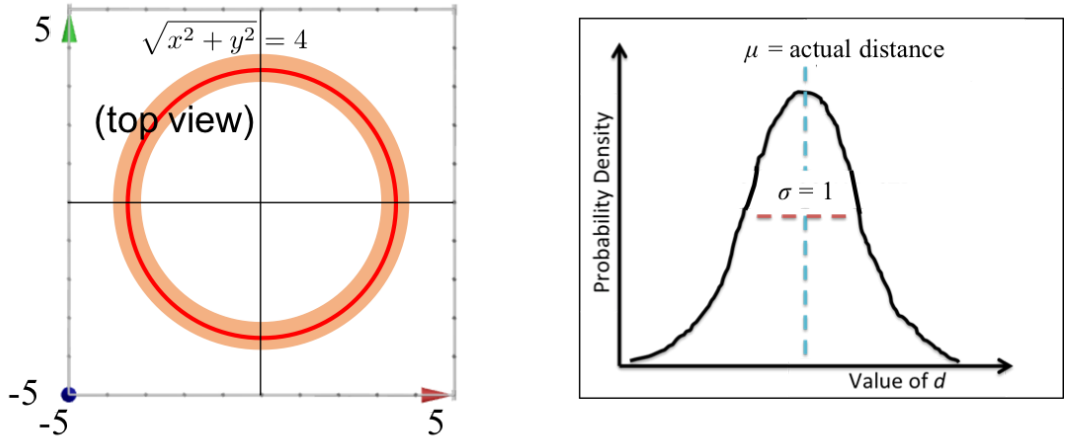
In this example we are going to explore the problem of tracking an object in 2D space. The object exists at some (x, y) location, however we are not sure exactly where! Thus we are going to use random variables X and Y to represent location. We have a prior belief about where the object is. In this example our prior both X and Y as normals which are independently distributed with mean 3 and variance 2. First let's write the prior belief as a joint probability density function

$$
\begin{aligned}
f(X = x, Y = y) &= f(X = x) \cdot f(Y = y) && \text{In the prior X and Y are independent} \\
&= \frac{1}{\sqrt{2 \cdot 4 \cdot \pi}} \cdot e^{-\frac{(x-3)^2}{2 \cdot 4}} \cdot \frac{1}{\sqrt{2 \cdot 4 \cdot \pi}} \cdot e^{-\frac{(y-3)^2}{2 \cdot 4}} && \text{Using the PDF equation for normals} \\
&= K_1 \cdot e^{-\frac{(x-3)^2 + (y-3)^2}{8}} && \text{All constants are put into } K_1
\end{aligned}
$$

This combinations of normals is called a bivariate distribution. Here is a visualization of the PDF of our prior.



The interesting part about tracking an object is the process of updating your belief about it's location based on an observation. Let's say that we get an instrument reading from a sonar that is sitting on the origin. The instrument reports that the object is 4 units away. Our instrument is not perfect: if the true distance was t units away, than the instrument will give a reading which is normally distributed with mean t and variance 1. Let's visualize the observation:



Based on this information about the noisiness of our prior, we can compute the conditional probability of seeing a particular distance reading D, given the true location of the object X, Y. If we knew the object was at location (x, y), we could calculate the true distance to the origin $\sqrt{x^2 + y^2}$ which would give us the mean for the instrument Gaussian:

$$
\begin{aligned}
f(D = d | X = x, Y = y) &= \frac{1}{\sqrt{2 \cdot 1 \cdot \pi}} \cdot e^{-\frac{(d - \sqrt{x^2+y^2})^2}{2 \cdot 1}} && \text{Normal PDF where } \mu = \sqrt{x^2 + y^2} \\
&= K_2 \cdot e^{-\frac{(d - \sqrt{x^2+y^2})^2}{2 \cdot 1}} && \text{All constants are put into } K_2
\end{aligned}
$$

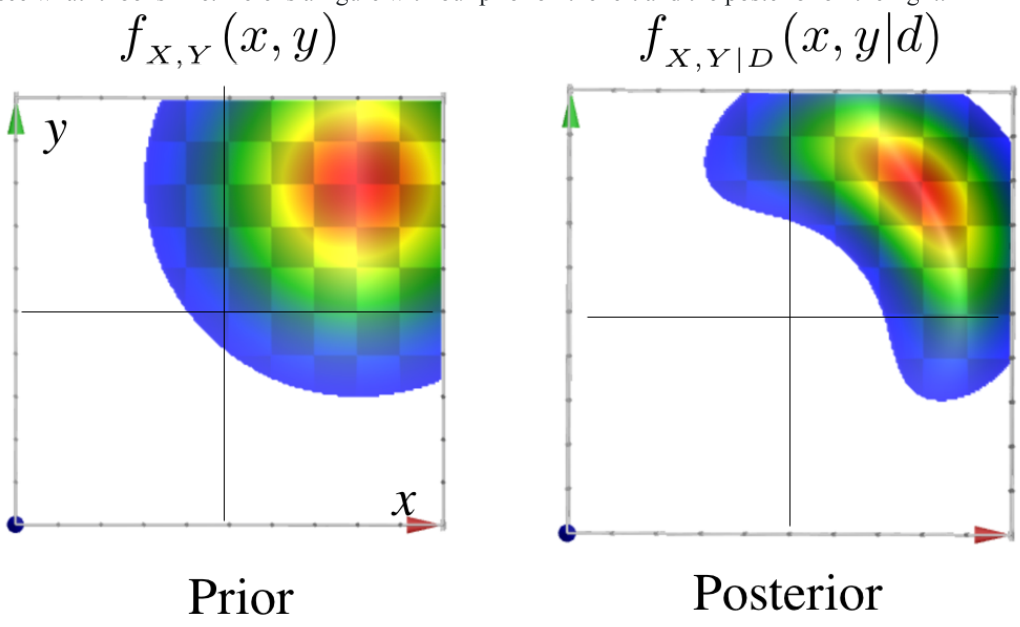
How about we try this out on actual numbers. How much more likely is an instrument reading of 1 compared to 2, given that the location of the object is at (1, 1)?

$$\begin{aligned}
\frac{f(D=1|X=1,Y=1)}{f(D=2|X=1,Y=1)} &= \frac{K_2 \cdot e^{-\frac{(1-\sqrt{1^2+1^2})^2}{2 \cdot 1}}}{K_2 \cdot e^{-\frac{(2-\sqrt{1^2+1^2})^2}{2 \cdot 1}}} && \text{Substituting into the conditional PDF of D} \\
&= \frac{e^0}{e^{-1/2}} \approx 1.65 && \text{Notice how the } K_2 \text{ cancel out}
\end{aligned}$$

At this point we have a prior belief and we have an observation. We would like to compute an updated belief, given that observation. This is a classic Bayes' formula scenario. We are using joint continuous variables, but that doesn't change the math much, it just means we will be dealing with densities instead of probabilities:

$$\begin{aligned}
& f(X=x, Y=y|D=4) \\
&= \frac{f(D=4|X=x,Y=y) \cdot f(X=x,Y=y)}{f(D=4)} && \text{Bayes using densities} \\
&= \frac{K_1 \cdot e^{-\frac{[4-\sqrt{x^2+y^2})^2]}{2}} \cdot K_2 \cdot e^{-\frac{[(x-3)^2+(y-3)^2]}{8}}}{f(D=4)} && \text{Substitute} \\
&= \frac{K_1 \cdot K_2}{f(D=4)} \cdot e^{-\left[\frac{[4-\sqrt{x^2+y^2})^2]}{2} + \frac{[(x-3)^2+(y-3)^2]}{8}\right]} && f(D=4) \text{ is a constant w.r.t. } (x,y) \\
&= K_3 \cdot e^{-\left[\frac{(4-\sqrt{x^2+y^2})^2}{2} + \frac{[(x-3)^2+(y-3)^2]}{8}\right]} && K_3 \text{ is a new constant}
\end{aligned}$$

Wow! That looks like a pretty interesting function! You have successfully computed the updated belief. Let's see what it looks like. Here is a figure with our prior on the left and the posterior on the right:



How beautiful is that! Its like a 2D normal distribution merged with a circle. But wait, what about that constant! We do not know the value of K_3 and that is not a problem for two reasons: the first reason is that if we ever want to calculate a relative probability of two locations, K_3 will cancel out. The second reason is that if we really wanted to know what K_3 was, we could solve for it. This math is used every day in millions of applications. If there are multiple observations the equations can get truly complex (even worse than this one). To represent these complex functions often use an algorithm called particle filtering.

Beta Distribution

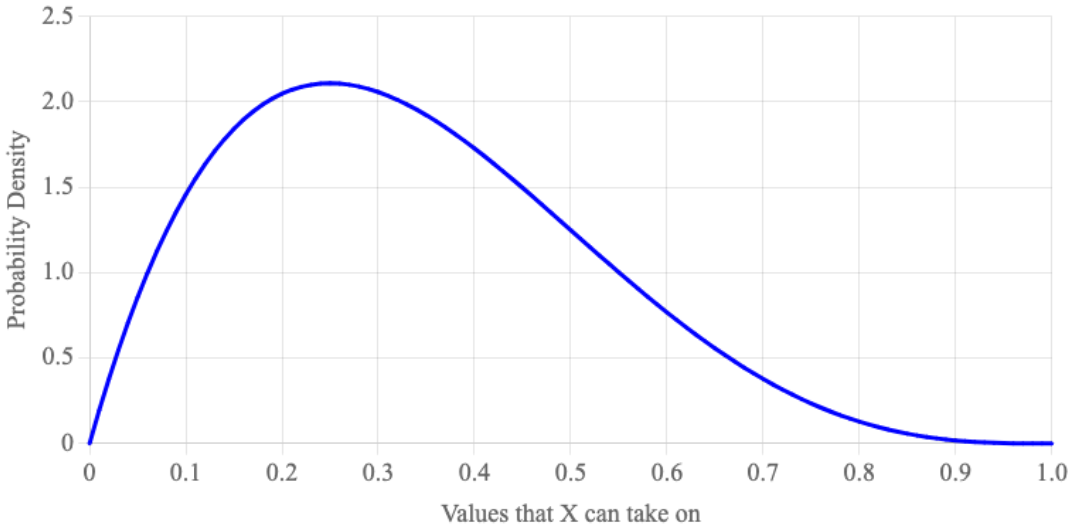
The Beta distribution is the distribution most often used as the distribution of probabilities. In this section we are going to have a very meta discussion about how we represent probabilities. Until now probabilities have just been numbers in the range 0 to 1. However, if we have uncertainty about our probability, it would make sense to represent our probabilities as random variables (and thus articulate the relative likelihood of our belief).

Beta Random Variable

Notation:	$X \sim \text{Beta}(a, b)$
Description:	A belief distribution over the value of a probability p from a Binomial distribution after observing $a-1$ successes and $b-1$ fails.
Parameters:	$a > 0$, the number successes + 1 $b > 0$, the number of fails + 1
Support:	$x \in [0, 1]$
PDF equation:	$f(x) = B \cdot x^{a-1} \cdot (1-x)^{b-1}$
CDF equation:	No closed form
Expectation:	$\text{E}[X] = \frac{a}{a+b}$
Variance:	$\text{Var}(X) = \frac{ab}{(a+b)^2(a+b+1)}$

PDF graph:

Parameter a: 2 Parameter b: 4



1. What is your Belief in p After 9 Heads in 10 Flips?

Imagine we have a coin and we would like to know its true probability of coming up heads, p. We flip the coin 10 times and observe 9 heads and 1 tail. What is your belief in p based off this evidence? Using the definition of probability we could guess that $p \approx \frac{9}{10}$. That number is a very rough estimate, especially since it is only based off 10 coin flips. Moreover the "point-value" $\frac{9}{10}$ does not have the ability to articulate how uncertain it is.

Could we instead have a random variable for the true probability? Formally, let X represent the true probability of the coin coming up heads. We don't use the symbol P for random variables, so X will have to do. If $X = 0.7$ then the probability of heads is 0.7. X must be a continuous random variable with support $[0, 1]$ since probabilities are continuous values which must be between 0 and 1.

Before flipping the coin, we could say that our belief about the coin's heads probability is uniform: $X \sim \text{Uni}(0, 1)$. Let H be a random variable for the number of heads and let T be a random variable for the number of tails observed. What is $\text{P}(X = x | H = 9, T = 1)$?

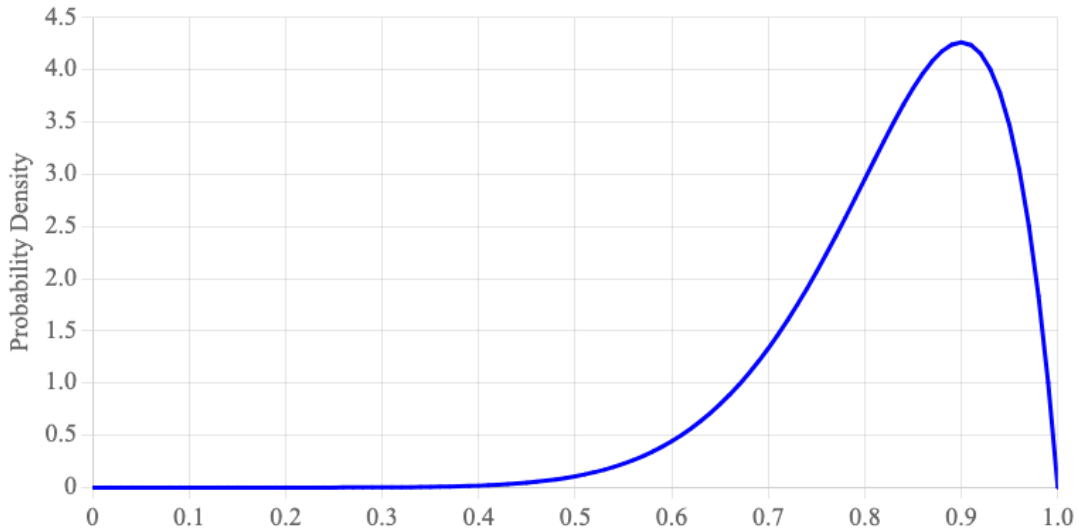
That probability is hard to think about! However it is much easier to reason about the probability with the condition reveresed: $P(H=9, T=1|X=x)$. This term asks the question: what is the probability of seeing 9 heads and 1 tail in 10 coin flips, given that the true probability of a heads is x. Convince yourself that this probability is just a binomial probability mass function with $n=10$ experiements, and $p=x$ evaluated at $k=9$ heads:

$$P(H=9, T=1|X=x) = \binom{10}{9} x^9 (1-x)^1$$

We are presented with a perfect context for Bayes' theorem with random variables. We know a conditional probability in one direction and we would like to know it in the other:

$$\begin{aligned}
& f(X=x|H=9, T=1) \\
&= \frac{P(H=9, T=1|X=x) \cdot f(X=x)}{P(H=9, T=1)} && \text{Bayes Theorem} \\
&= \frac{\binom{10}{9} x^9 (1-x)^1 \cdot f(X=x)}{P(H=9, T=1)} && \text{Binomial PMF} \\
&= \frac{\binom{10}{9} x^9 (1-x)^1 \cdot 1}{P(H=9, T=1)} && \text{Uniform PDF} \\
&= \frac{\binom{10}{9}}{P(H=9, T=1)} x^9 (1-x)^1 && \text{Constants to front} \\
&= K \cdot x^9 (1-x)^1 && \text{Rename constant}
\end{aligned}$$

Lets take a look at that function. For now we can let $K = \frac{1}{110}$. Regardless of K we will get the same shape, just scaled:



What a beautiful image. It tells us relatively likelihood over the probability that is governing our coinflips. Here are a few observations from this chart:

1. Even after only 10 coin flips we are very confident that the true probability is > 0.5
2. It is almost 10 times more likely that $X = 0.9$ as it is that $X = 0.6$.
3. $f(X=1) = 0$, which makes sense. How could we have flipped that one tail if the probability of heads was 1?

Wait but why?

In the derivation above for $f(X=x|H=9, T=1)$ we made the claim that $P(H=9, T=1)$ is a constant. A lot of folks find that hard to believe. Why is that the case?

It may be helpful to juxtapose $P(H=9, T=1)$ with $P(H=9, T=1|X=x)$. The later says "what is the probability of 9 heads, given the true probability is x". The former says "what is the probability of 9 heads, under all possible assignments of x". If you wanted to calculate $P(H=9, T=1)$ you could use the law of total probability:

$$\begin{aligned}
& P(H=9, T=1) \\
&= \int_{y=0}^{1} P(H=9, T=1|X=y) f(X=y)
\end{aligned}$$

That is a hard number to calculate, but it is in fact a constant with respect to x.

2. Beta Derivation

Let's repeat the derivation from the previous section, using non-random variables for the number of observed heads, h, and the number of tails, t, observed.

If we let $H = h$ be the event that we saw h heads, and let $T = t$ be the even that we saw t tails in $h + t$ coinflips. We want to calculate the probability density function $f(X = x|H = h, T = t)$. We can use the exam same series of steps, starting with Bayes Theorem:

$$
\begin{aligned}
& f(X = x|H = h, T = t) \\
&= \frac{P(H = h, T = t|X = x)f(X = x)}{P(H = h, T = t)} && \text{Bayes Theorem} \\
&= \frac{\binom{h+t}{h}x^n(1-x)^h}{P(H = h, T = t)} && \text{Binomial PMF, Uniform PDF} \\
&= \frac{\binom{h+t}{h}}{P(H = h, T = t)}x^h(1-x)^t && \text{Moving terms around} \\
&= \frac{1}{c} \cdot x^h(1-x)^t && \text{where } c = \int_0^1 x^h(1-x)^t dx
\end{aligned}
$$

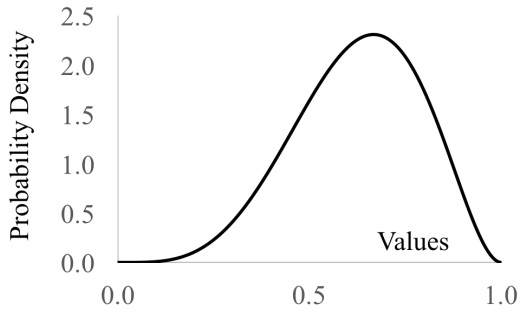
The equation that we arrived at when using a Bayesian approach to estimating our probability defines a probability density function and thus a random variable. The random variable is called a Beta distribution, and it is defined as follows:

The Probability Density Function (PDF) for $X \sim \text{Beta}(a, b)$ is:

$$
f(X = x) = \begin{cases} \frac{1}{B(a,b)}x^{a-1}(1-x)^{b-1} & \text{if } 0 < x < 1 \\ 0 & \text{otherwise} \end{cases} \qquad \text{where } B(a, b) = \int_0^1 x^{a-1}(1-x)^{b-1}dx
$$

A Beta distribution has $E[X] = \frac{a}{a+b}$ and $Var(X) = \frac{ab}{(a+b)^2(a+b+1)}$. All modern programming languages have a package for calculating Beta CDFs. You will not be expected to compute the CDF by hand in CS109.

To model our estimate of the probability of a coin coming up heads: set $a = h + 1$ and $b = t + 1$. Beta is used as a random variable to represent a belief distribution of probabilities in contexts beyond estimating coin flips. For example perhaps a drug has been given to 6 patients, 4 of whom have been cured. We could express our belief in the probability that the drug can cure patients as $X \sim \text{Beta}(a = 5, b = 3)$:



Notice how the most likely belief for the probability of curing a patient, is $4/6$, the fraction of patients cured. This distribution shows that we hold a non-zero belief that the probability could be something other than $4/6$. It is unlikely that the probability is 0.01 or 0.09, but reasonably likely that it could be 0.5.

3. Beta as a Prior

You can set $X \sim \text{Beta}(a, b)$ as a prior to reflect how biased you think the coin is apriori to flipping it. This is a subjective judgment that represent $a + b - 2$ "imaginary" trials with $a - 1$ heads and $b - 1$ tails. If you then observe $h + t$ real trials with h heads you can update your belief. Your new belief would be, $X \sim \text{Beta}(a + h, b + t)$. Using the prior $\text{Beta}(1, 1) = \text{Uni}(0, 1)$ is the same as saying we haven't seen any "imaginary" trials, so apriori we know nothing about the coin. Here is the proof for the distribution of X when the prior was a Beta too:

If our prior belief for $X \sim \text{Beta}(a, b)$, then our posterior is $\text{Beta}(a + h, b + t)$:

$$
\begin{aligned}
&f(X = x | H = h, T = t) \\
&= \frac{P(H = h, T = t | X = x) f(X = x)}{P(H = h, T = t)} && \text{Bayes Theorem} \\
&= \frac{\binom{h+t}{h} x^h (1-x)^t \cdot \frac{1}{c} \cdot x^{a-1}(1-x)^{b-1}}{P(H = h, T = t)} && \text{Beta PMF, Uniform PDF} \\
&= K \cdot x^h (1-x)^t \cdot x^{a-1}(1-x)^{b-1} && \text{Combine Constants} \\
&= K \cdot x^{a+h-1}(1-x)^{b+t-1} && \text{Combine Like Bases}
\end{aligned}
$$

Which is the PDF of $\text{Beta}(a+h, b+t)$

It is pretty convenient that if we have a Beta prior belief, then our posterior belief is also Beta. This makes Betas especially convenient to work with, in code and in proof, if there are many updates that you will make to your belief over time. This property where the type of distribution is the same before and after an observation is called a conjugate prior.

Quick question: Are you allowed to just make up priors and imaginary trials? Some folks think that is fine (they are called Bayesians) and some folks think that you shouldn't make up prior beliefs (they are called frequentists). In general, for small data it can make you much better at making predictions if you are able to come up with a good prior belief.

Observation: There is a deep connection between the beta-prior and the uniform-prior (which we used initially). It turns out that $\text{Beta}(1,1) = \text{Uni}(0,1)$. Recall that $\text{Beta}(1,1)$ means 0 imaginary heads and 0 imaginary tails.

Adding Random Variables

In this section on uncertainty theory we are going to explore some of the great results in probability theory. As a gentle introduction we are going to start with convolution. Convolution is a very fancy way of saying "adding" two different random variables together. The name comes from the fact that adding two random varaibles requires you to "convolve" their distribution functions. It is interesting to study in detail because (1) many natural processes can be modelled as the sum of random variables, and (2) because mathemeticians have made great progress on proving convolution theorems. For some particular random variables computing convolution has closed form equations. Importantly convolution is the sum of the random variables themselves, not the addition of the probability density functions (PDF)s that correspond to the random variables.

1. Adding Two Random Variables

Deriving an expression for the likelihood for the sum of two random variables requires an interesting insight. If your random variables are discrete then the probability that $X + Y = n$ is the sum of mutually exclusive cases where X takes on a values in the range $[0, n]$ and Y takes on a value that allows the two to sum to n. Here are a few examples $X = 0 \text{ and } Y = n$, $X = 1 \text{ and } Y = n - 1$ etc. In fact all of the mutually exclusive cases can be enumerated in a sum:

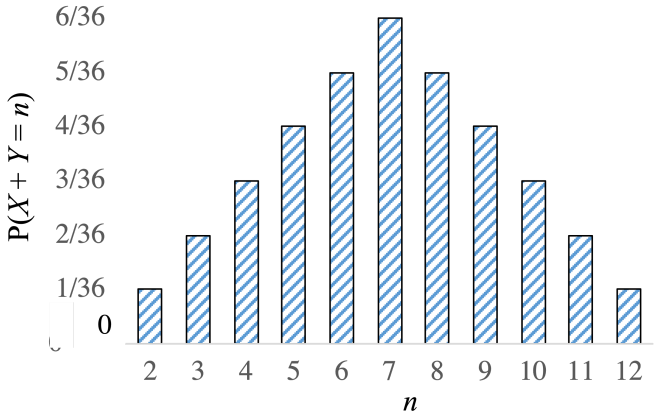
Def: General Rule for the Convolution of Discrete Variables

$$\mathrm{P}(X + Y = n) = \sum_{i=-\infty}^{\infty} \mathrm{P}(X = i, Y = n - i)$$

If the random variables are independent you can futher decompose the term $\mathrm{P}(X = i, Y = n - i)$. Let's expand on some of the mutually exclusive cases where $X + Y = n$:

i	X	Y	
0	0	n	$\mathrm{P}(X = 0, Y = n)$
1	1	$n - 1$	$\mathrm{P}(X = 1, Y = n - 1)$
2	2	$n - 2$	$\mathrm{P}(X = 2, Y = n - 2)$
	...		
n	n	0	$\mathrm{P}(X = n, Y = 0)$

Consider the sum of two independent dice. Let X and Y be the outcome of each dice. Here is the probability mass function for the sum $X + Y$:



Let's use this context to practice deriving the sum of two variables, in this case $\mathrm{P}(X+Y=n)$, starting with the General Rule for the Convolution of Discrete Random Variables. We start by considering values of n between 2 and 7. In this range $\mathrm{P}(X=i, Y=n-i) = \frac{1}{36}$ for all values of i between 1 and $n-1$. There is exactly one outcome of the two die where $X=i$ and $Y=n-i$. For values of i outside this range $n-i$ is not a valid dice outcome and $\mathrm{P}(X=i, Y=n-i) = 0$:

$$\begin{aligned}
&\mathrm{P}(X+Y=n) \\
&= \sum_{i=-\infty}^{\infty} \mathrm{P}(X=i, Y=n-i) \\
&= \sum_{i=1}^{n-1} \mathrm{P}(X=i, Y=n-i) \\
&= \sum_{i=1}^{n-1} \frac{1}{36} \\
&= \frac{n-1}{36}
\end{aligned}$$

For values of n greater than 7 we could use the same approach, though different values of i would make $\mathrm{P}(X=i, Y=n-i)$ non-zero.

This derivation for a general rule has a continuous equivalent:

$$f(X+Y=n) = \int_{i=-\infty}^{\infty} f(X=n-i, Y=i)\, di$$

2. Sum of Independent Poissons

For any two Poisson random variables: $X \sim \mathrm{Poi}(\lambda_1)$ and $Y \sim \mathrm{Poi}(\lambda_2)$ the sum of those two random variables is another Poisson: $X+Y \sim \mathrm{Poi}(\lambda_1+\lambda_2)$. This holds even when λ_1 is not the same as λ_2.

How could we prove a the above claim?

Example derivation:

Let's go about proving that the sum of two independent Poisson random variables is also Poisson. Let $X \sim \mathrm{Poi}(\lambda_1)$ and $Y \sim \mathrm{Poi}(\lambda_2)$ be two independent random variables, and $Z = X+Y$. What is $P(Z=n)$?

$$
\begin{aligned}
P(Z=n) &= P(X+Y=n) \\
&= \sum_{k=-\infty}^{\infty} \mathrm{P}(X=k, Y=n-k) && \text{(Convolution)} \\
&= \sum_{k=-\infty}^{\infty} P(X=k)P(Y=n-k) && \text{(Independence)} \\
&= \sum_{k=0}^{n} P(X=k)P(Y=n-k) && \text{(Range of } X \text{ and } Y\text{)} \\
&= \sum_{k=0}^{n} e^{-\lambda_1}\frac{\lambda_1^k}{k!}e^{-\lambda_2}\frac{\lambda_2^{n-k}}{(n-k)!} && \text{(Poisson PMF)} \\
&= e^{-(\lambda_1+\lambda_2)}\sum_{k=0}^{n}\frac{\lambda_1^k\lambda_2^{n-k}}{k!(n-k)!} \\
&= \frac{e^{-(\lambda_1+\lambda_2)}}{n!}\sum_{k=0}^{n}\frac{n!}{k!(n-k)!}\lambda_1^k\lambda_2^{n-k} \\
&= \frac{e^{-(\lambda_1+\lambda_2)}}{n!}(\lambda_1+\lambda_2)^n && \text{(Binomial theorem)}
\end{aligned}
$$

Note that the Binomial Theorem (which we did not cover in this class, but is often used in contexts like expanding polynomials) says that for two numbers a and b and positive integer n, $(a+b)^n = \sum_{k=0}^{n}\binom{n}{k}a^k b^{n-k}$.

3. Sum of Independent Binomials with equal p

For any two independent Binomial random variables with the same "success" probability p:
$X \sim \text{Bin}(n_1, p)$ and $Y \sim \text{Bin}(n_2, p)$ the sum of those two random variables is another binomial:
$X + Y \sim \text{Bin}(n_1 + n_2, p)$.

This result hopefully makes sense. The convolution is the number of sucesses across X and Y. Since each trial has the same probability of success, and there are now $n_1 + n_2$ trials, which are all independent, the convolution is simply a new Binomial. This rule does not hold when the two Binomial random variables have different parameters p.

4. Sum of Independent Normals

For any two independent normal random variables $X \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $Y \sim \mathcal{N}(\mu_2, \sigma_2^2)$ the sum of those two random variables is another normal: $X + Y \sim \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$.

Again this only holds when the two normals are independent.

5. Sum of Independent Uniforms

If X and Y are independent uniform random variables where $X \sim \text{Uni}(0, 1)$ and $Y \sim \text{Uni}(0, 1)$:

$$
f(X+Y=n) = \begin{cases} n & \text{if } 0 < n \leq 1 \\ 2-n & \text{if } 1 < n \leq 2 \\ 0 & \text{else} \end{cases}
$$

Example derivation:
Calculate the PDF of $X + Y$ for independent uniform random variables $X \sim \text{Uni}(0, 1)$ and $Y \sim \text{Uni}(0, 1)$? First plug in the equation for general convolution of independent random variables:

$$
\begin{aligned}
f(X+Y=n) &= \int_{i=0}^{1} f(X=n-i, Y=i)di \\
&= \int_{i=0}^{1} f(X=n-i)f(Y=i)di && \text{Independence} \\
&= \int_{i=0}^{1} f(X=n-i)di && \text{Because } f(Y=y)=1
\end{aligned}
$$

It turns out that is not the easiest thing to integrate. By trying a few different values of n in the range $[0, 2]$ we can observe that the PDF we are trying to calculate is discontinuous at the point $n = 1$ and thus will be easier to think about as two cases: $n < 1$ and $n > 1$. If we calculate $f(X + Y = n)$ for both cases and correctly constrain the bounds of the integral we get simple closed forms for each case:

$$f(X + Y = n) = \begin{cases} n & \text{if } 0 < n \leq 1 \\ 2 - n & \text{if } 1 < n \leq 2 \\ 0 & \text{else} \end{cases}$$

Central Limit Theorem

There are two ways that you could state the central limit theorem. Either that the sum of IID random variables is normally distributed, or that the average of IID random variables is normally distributed.

The Central Limit Thorem (Sum Version)
Let $X_1, X_2 \dots X_n$ be independent and identically distributed random variables. The **sum** of these random variables approaches a normal as $n \to \infty$:

$$\sum_{i=1}^{n} X_i \sim N(n \cdot \mu, n \cdot \sigma^2)$$

Where $\mu = \mathrm{E}[X_i]$ and $\sigma^2 = \mathrm{Var}(X_i)$. Note that since each X_i is identically distributed they share the same expectation and variance.

At this point you probably think that the central limit theorem is awesome. But it gets even better. With some algebraic manipulation we can show that if the sample mean of IID random variables is normal, it follows that the sum of equally weighted IID random variables must also be normal:

The Central Limit Thorem (Average Version)
Let $X_1, X_2 \dots X_n$ be independent and identically distributed random variables. The **average** of these random variables approaches a normal as $n \to \infty$:

$$\frac{1}{n}\sum_{i=1}^{n} X_i \sim N(\mu, \frac{\sigma^2}{n})$$

Where $\mu = \mathrm{E}[X_i]$ and $\sigma^2 = \mathrm{Var}(X_i)$.

1. Central Limit Theorem Intuition

In the previous section we explored what happens when you add two random variables. What happens when you add more than two random variables? For example, what if I wanted to add up 100 different uniform random variables:

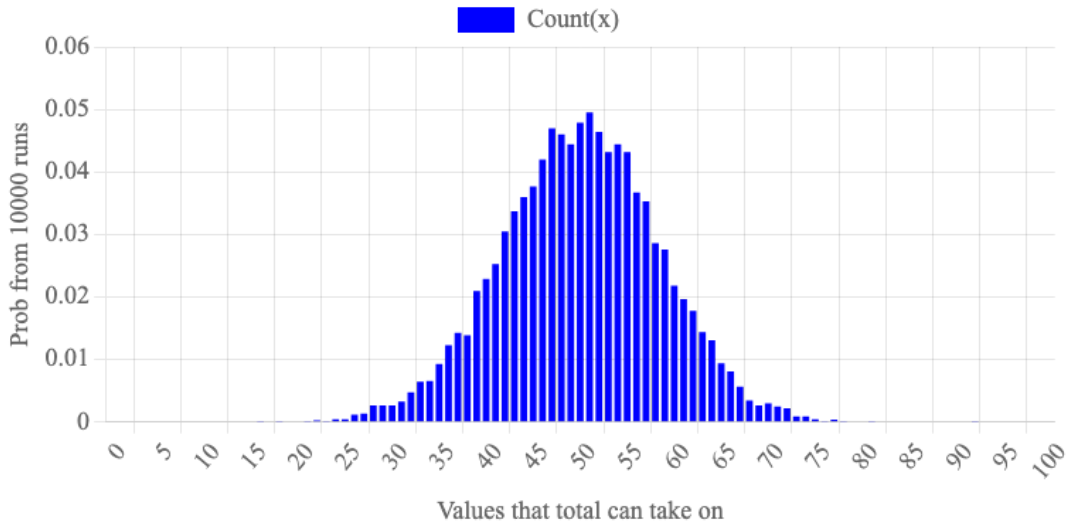
```
from random import random

def add_100_uniforms():
    total = 0
    for i in range(100):
        # returns a sample from uniform(0, 1)
        x_i = random()
        total += x_i
    return total
```

The value, `total` returned by this function will be a random variable. Hit the button below to run the function and observe the resulting value of total:

`add_100_uniforms()` `total: 49.32722`

What does total look like as a distribution? Let's calculate `total` many times and visualize the histogram of values it produces.



10,000 more runs

That is interesting! **`total`** which is the sum of 100 independent uniforms looks normal. Is that a special property of uniforms? No! It turns out to work for almost any type of distribution (as long as the thing you are adding has finite mean and finite variance, everything we have covered in this reader).

- Sum of 40 X_i where $X_i \sim \text{Beta}(a = 5, b = 4)$? Normal.
- Sum of 90 X_i where $X_i \sim \text{Poi}(\lambda = 4)$? Normal.
- Sum of 50 dice-rolls? Normal.
- Average of 10000 X_i where $X_i \sim \text{Exp}(\lambda = 8)$? Normal.

For any distribution the sum, or average, of n independent equally-weighted samples from that distribution, will be normal.

2. Continuity Correction

Now we can see that the Binomial Approximation using a Normal actually derives from the central limit theorem. Recall that, when computing probabilities for a normal approximation, we had to to use a continuity correction. This was because we were approximating a discrete random variable (a binomial) with a continuous one (a normal). You should use a continuity correction any time your normal is approximating a discrete random variable. The rules for a general continuity correction are the same as the rules for the binomial-approximation continuity correction.

In the motivating example above, where we added 100 uniforms, a continuity correction isn't needed because the sum of uniforms is continuous. In the dice sum example below, a continuity correction is needed because die outcomes are discrete.

3. Examples

Example:

You will roll a 6 sided dice 10 times. Let X be the total value of all 10 dice = $X_1 + X_2 + \cdots + X_{10}$. You win the game if $X \leq 25$ or $X \geq 45$. Use the central limit theorem to calculate the probability that you win. Recall that $E[X_i] = 3.5$ and $\text{Var}(X_i) = \frac{35}{12}$.

Let Y be the approximating normal. By the Central Limit Theorem $Y \sim N(10 \cdot E[X_i], 10 \cdot \text{Var}(X_i))$. Substituting in the known values for expectation and variance: $Y \sim N(35, 29.2)$

$$
\begin{aligned}
&\text{P}(X \leq 25 \text{ or } X \geq 45) \\
&\quad = \text{P}(X \leq 25) + \text{P}(X \geq 45) \\
&\quad \approx \text{P}(Y < 25.5) + \text{P}(Y > 44.5) && \text{Continuity Correction} \\
&\quad \approx \text{P}(Y < 25.5) + [1 - \text{P}(Y < 44.5)] \\
&\quad \approx \Phi(\frac{25.5 - 35}{\sqrt{29.2}}) + \left[1 - \Phi(\frac{44.5 - 35}{\sqrt{29.2}})\right] && \text{Normal CDF} \\
&\quad \approx \Phi(-1.76) + [1 - \Phi(1.76)] \\
&\quad \approx 0.039 + (1 - 0.961) \approx 0.078
\end{aligned}
$$

Example:

Say you have a new algorithm and you want to test its running time. You have an idea of the variance of the algorithm's run time: $\sigma^2 = 4\text{sec}^2$ but you want to estimate the mean: $\mu = t\text{sec}$. You can run the algorithm repeatedly (IID trials). How many trials do you have to run so that your estimated runtime = $t \pm 0.5$ with 95\% certainty? LetX_i be the run time of the i-th run (for $1 \leq i \leq n$).

$$0.95 = P(-0.5 \leq \frac{\sum_{i=1}^{n} X_i}{n} - t \leq 0.5)$$

By the central limit theorem, the standard normal Z must be equal to:

$$\begin{aligned} Z &= \frac{\left(\sum_{i=1}^{n} X_i\right) - n\mu}{\sigma\sqrt{n}} \\ &= \frac{\left(\sum_{i=1}^{n} X_i\right) - nt}{2\sqrt{n}} \end{aligned}$$

Now we rewrite our probability inequality so that the central term is Z:

$$\begin{aligned} 0.95 &= P(-0.5 \leq \frac{\sum_{i=1}^{n} X_i}{n} - t \leq 0.5) = P(\frac{-0.5\sqrt{n}}{2} \leq \frac{\sum_{i=1}^{n} X_i}{n} - t \leq \frac{0.5\sqrt{n}}{2}) \\ &= P(\frac{-0.5\sqrt{n}}{2} \leq \frac{\sqrt{n}}{2}\frac{\sum_{i=1}^{n} X_i}{n} - \frac{\sqrt{n}}{2}t \leq \frac{0.5\sqrt{n}}{2}) = P(\frac{-0.5\sqrt{n}}{2} \leq \frac{\sum_{i=1}^{n} X_i}{2\sqrt{n}} - \frac{\sqrt{n}}{\sqrt{n}}\frac{\sqrt{n}t}{2} \leq \\ &= P(\frac{-0.5\sqrt{n}}{2} \leq \frac{\sum_{i=1}^{n} X_i - nt}{2\sqrt{n}} \leq \frac{0.5\sqrt{n}}{2}) \\ &= P(\frac{-0.5\sqrt{n}}{2} \leq Z \leq \frac{0.5\sqrt{n}}{2}) \end{aligned}$$

And now we can find the value of n that makes this equation hold.

$$\begin{aligned} 0.95 &= \phi(\frac{\sqrt{n}}{4}) - \phi(-\frac{\sqrt{n}}{4}) = \phi(\frac{\sqrt{n}}{4}) - (1 - \phi(\frac{\sqrt{n}}{4})) \\ &= 2\phi(\frac{\sqrt{n}}{4}) - 1 \\ 0.975 &= \phi(\frac{\sqrt{n}}{4}) \\ \phi^{-1}(0.975) &= \frac{\sqrt{n}}{4} \\ 1.96 &= \frac{\sqrt{n}}{4} \\ n &= 61.4 \end{aligned}$$

Thus it takes 62 runs. If you are interested in how this extends to cases where the variance is unknown, look into variations of the students' t-test.

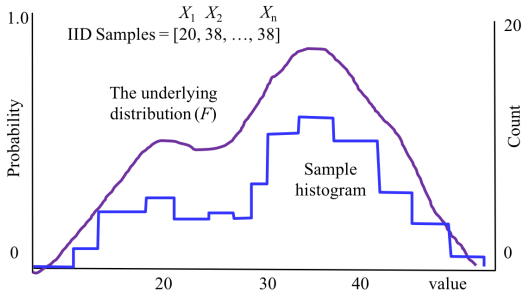
Sampling

In this section we are going to talk about statistics calculated on samples from a population. We are then going to talk about probability claims that we can make with respect to the original population -- a central requirement for most scientific disciplines.

Let's say you are the king of Bhutan and you want to know the average happiness of the people in your country. You can't ask every single person, but you could ask a random subsample. In this next section we will consider principled claims that you can make based on a subsample. Assume we randomly sample 200 Bhutanese and ask them about their happiness. Our data looks like this: $72, 85, \ldots, 71$. You can also think of it as a collection of $n = 200$ I.I.D. (independent, identically distributed) random variables $X_1, X_2, \ldots, X_n$.

1. Understanding Samples

The idea behind sampling is simple, but the details and the mathematical notation can be complicated. Here is a picture to show you all of the ideas involved:



The theory is that there is some large population (for example the 774,000 people who live in Bhutan). We collect a sample of n people at random, where each person in the population is equally likely to be in our sample. From each person we record one number (for example their reported happiness). We are going to call the number from the ith person we sampled X_i. One way to visualize your samples $X_1, X_2, \ldots, X_n$ is to make a histogram of their values.

We make the assumption that all of our X_is are identically distributed. That means that we are assuming there is a single underlying distribution F that we drew our samples from. Recall that a distribution for discrete random variables should define a probability mass function.

2. Estimating Mean and Variance from Samples

We assume that the data we look at are IID from the same underlying distribution (F) with a true mean (μ) and a true variance (σ^2). Since we can't talk to everyone in Bhutan we have to rely on our sample to estimate the mean and variance. From our sample we can calculate a sample mean ($\bar{X}$) and a sample variance (S^2). These are the best guesses that we can make about the true mean and true variance.

$$\bar{X} = \sum_{i=1}^{n} \frac{X_i}{n} \qquad S^2 = \sum_{i=1}^{n} \frac{(X_i - \bar{X})^2}{n-1}$$

The first question to ask is, are those unbiased estimates? Yes. Unbiased, means that if we were to repeat this sampling process many times, the expected value of our estimates should be equal to the true values we are trying to estimate. We will prove that that is the case for $\bar{X}$. The proof for S^2 is in lecture slides.

$$E[\bar{X}] = E[\sum_{i=1}^{n} \frac{X_i}{n}] = \frac{1}{n} E\left[\sum_{i=1}^{n} X_i\right]$$
$$= \frac{1}{n} \sum_{i=1}^{n} E[X_i] = \frac{1}{n} \sum_{i=1}^{n} \mu = \frac{1}{n} n\mu = \mu$$

The equation for sample mean seems related to our understanding of expectation. The same could be said about sample variance except for the surprising $(n-1)$ in the denominator of the equation. Why $(n-1)$? That denominator is necessary to make sure that the $E[S^2] = \sigma^2$.

The intuition behind the proof is that sample variance calculates the distance of each sample to the sample mean, \emph{not} the true mean. The sample mean itself varies, and we can show that its variance is also related to the true variance.

3. Standard Error

Ok, you convinced me that our estimates for mean and variance are not biased. But now I want to know how much my sample mean might vary relative to the true mean.

$$\begin{aligned}
\text{Var}(\bar{X}) &= \text{Var}(\sum_{i=1}^{n} \frac{X_i}{n}) = \left(\frac{1}{n}\right)^2 \text{Var}\left(\sum_{i=1}^{n} X_i\right) \\
&= \left(\frac{1}{n}\right)^2 \sum_{i=1}^{n} \text{Var}(X_i) = \left(\frac{1}{n}\right)^2 \sum_{i=1}^{n} \sigma^2 = \left(\frac{1}{n}\right)^2 n\sigma^2 = \frac{\sigma^2}{n} \\
&\approx \frac{S^2}{n} \\
\text{Std}(\bar{X}) &\approx \sqrt{\frac{S^2}{n}}
\end{aligned}$$

That term, Std($\bar{X}$), has a special name. It is called the standard error and its how you report uncertainty of estimates of means in scientific papers (and how you get error bars). Great! Now we can compute all these wonderful statistics for the Bhutanese people. But wait! You never told me how to calculate the Std(S^2). That is hard because the central limit theorem doesn't apply to the computation of S^2. Instead we will need a more general technique. See the next chapter: Bootstrapping

Let's say we calculate the our sample of happiness has n = 200 people. The sample mean is $\bar{X} = 83$ (what is the unit here? happiness score?) and the sample variance is $S^2 = 450$. We can now calculate the standard error of our estimate of the mean to be 1.5. When we report our results we will say that our estimate of the average happiness score in Bhutan is 83 $\pm$ 1.5. Our estimate of the variance of happiness is 450 $\pm$?.

Bootstrapping

The bootstrap is a newly invented statistical technique for both understanding distributions of statistics and for calculating p-values (a p-value is a the probability that a scientific claim is incorrect). It was invented here at Stanford in 1979 when mathematicians were just starting to understand how computers, and computer simulations, could be used to better understand probabilities.

The first key insight is that: if we had access to the underlying distribution (F) then answering almost any question we might have as to how accurate our statistics are becomes straightforward. For example, in the previous section we gave a formula for how you could calculate the sample variance from a sample of size n. We know that in expectation our sample variance is equal to the true variance. But what if we want to know the probability that the true variance is within a certain range of the number we calculated? That question might sound dry, but it is critical to evaluating scientific claims! If you knew the underlying distribution, F, you could simply repeat the experiment of drawing a sample of size n from F , calculate the sample variance from our new sample and test what portion fell within a certain range.

The next insight behind bootstrapping is that the best estimate that we can get for F is from our sample itself! The simplest way to estimate F (and the one we will use in this class) is to assume that the $P(X = k)$ is simply the fraction of times that k showed up in the sample. Note that this defines the probability mass function of our estimate $\hat{F}$ of F.

```
def bootstrap(sample):
   N = number of elements in sample
   pmf = estimate the underlying pmf from the sample
   stats = []
   repeat 10,000 times:
      resample = draw N new samples from the pmf
      stat = calculate your stat on the resample
      stats.append(stat)
   stats can now be used to estimate the distribution of the stat
```

Bootstrapping is a reasonable thing to do because the sample you have is the best and only information you have about what the underlying population distribution actually looks like. Moreover most samples will, if they're randomly chosen, look quite like the population they came from.

To calculate $\text{Var}(S^2)$ we could calculate S_i^2 for each resample i and after 10,000 iterations, we could calculate the sample variance of all the S_i^2s. You might be wondering why the resample is the same size as the original sample (n). The answer is that the variation of the variation of stat that you are calculating could depend on the size of the sample (or the resample). To accurately estimate the distribution of the stat we must use resamples of the same size.

The bootstrap has strong theoretic grantees, and is accepted by the scientific community. It breaks down when the underlying distribution has a ``long tail'' or if the samples are not I.I.D.

1. Example of p-value calculation

We are trying to figure out if people are happier in Bhutan or in Nepal. We sample $n_1 = 200$ individuals in Bhutan and $n_2 = 300$ individuals in Nepal and ask them to rate their happiness on a scale from 1 to 10. We measure the sample means for the two samples and observe that people in Nepal are slightly happier--the difference between the Nepal sample mean and the Bhutan sample mean is 0.5 points on the happiness scale.

If you want to make this claim scientific you should calculate a p-value. A p-value is the probability that, when the null hypothesis is true, the statistic measured would be equal to, or more extreme than, than the value you are reporting. The null hypothesis is the hypothesis that there is no relationship between two measured phenomena or no difference between two groups.

In the case of comparing Nepal to Bhutan, the null hypothesis is that there is no difference between the distribution of happiness in Bhutan and Nepal. The null hypothesis argument is: there is no difference in the distribution of happiness between Nepal and Bhutan. When you drew samples, Nepal had a mean that 0.5 points larger than Bhutan by chance.

We can use bootstrapping to calculate the p-value. First, we estimate the underlying distribution of the null hypothesis underlying distribution, by making a probability mass function from all of our samples from Nepal and all of our samples from Bhutan.

```
def pvalue_bootstrap(bhutan_sample, nepal_sample):
   N = size of the bhutan_sample
   M = size of the nepal_sample
   universal_sample = combine bhutan_samples and nepal_samples
   universal_pmf = estimate the underlying pmf of the universalSample
   count = 0
   observed_difference = mean(nepal_sample) - mean(bhutan_sample)
   repeat 10,000 times:
      bhutan_resample = draw N new samples from the universalPmf
      nepal_resample = draw M new samples from the universalPmf
      mu_bhutan = sample mean of the bhutanResample
      mu_nepal = sample mean of the nepalResample
      mean_difference = |muNepal - muBhutan|
      if mean_difference > observed_difference:
         count += 1
   pvalue = count / 10,000
```

This is particularly nice because nowhere did we have to make an assumption about a parametric distribution that our samples came from (ie we never had to claim that happiness is gaussian). You might have heard of a t-test. That is another way of calculating p-values, but it makes the assumption that both samples are gaussian and that they both have the same variance. In the modern context where we have reasonable computer power, bootstrapping is a more correct and versatile tool.

Algorithmic Analysis

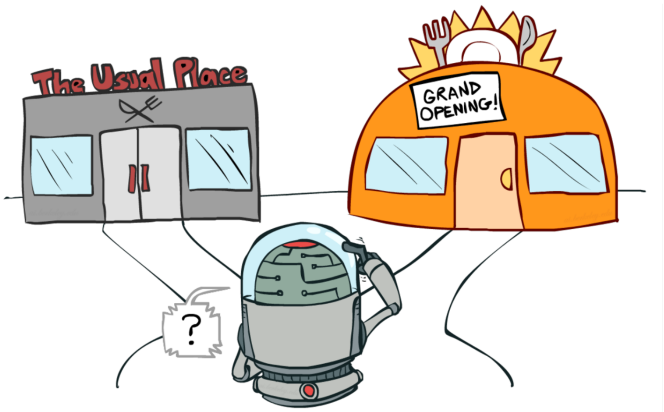
In this section we are going to use probability to analyze code. Specifically we are going to be calculating expectations on code: expected run time, expected resulting values etc. The reason that we are going to focus on expectation is that it has several nice properties. One of the most useful properties that we have seen so far is that the expectation of a sum, is the sum of expectations, *regardless* of whether the random variables are independent of one another. In this section we will see a few more helpful properties, including the Law of Total Expectation, which is also helpful in analyzing code:

Law of Total Expectation

The law of total expectation gives you a way to calculate $\mathrm{E}[X]$ in the scenareo where it is easier to compute $\mathrm{E}[X|Y=y]$ where Y is some other random variable:

$$\begin{aligned}\mathrm{E}[X] &= \mathrm{E}\left[E[X|Y]\right] \\ &= \sum_{y} \mathrm{E}[X|Y=y]\,\mathrm{P}(Y=y)\end{aligned}$$

Thompson Sampling



Warning: This chapter is a stub. Come back later and hopefully someone has had a chance to finish it.

Let me present you with a seemingly simple problem that has a suprisingly complex solution. Imagine that you have two brand new drugs for a serious illness. You don't know how effective each drug is. You want to know which drug is the most effective, but at the same time, there are costs to *exploration* — there are high stakes.

Parameter Estimation

We have learned many different distributions for random variables and all of those distributions had parameters: the numbers that you provide as input when you define a random variable. So far when we were working with random variables, we either were explicitly told the values of the parameters, or, we could divine the values by understanding the process that was generating the random variables.

What if we don't know the values of the parameters and we can't estimate them from our own expert knowledge? What if instead of knowing the random variables, we have a lot of examples of data generated with the same underlying distribution? In this chapter we are going to learn formal ways of estimating parameters from data.

These ideas are critical for artificial intelligence. Almost all modern machine learning algorithms work like this: (1) specify a probabilistic model that has parameters. (2) Learn the value of those parameters from data.

1. Parameters

Before we dive into parameter estimation, first let's revisit the concept of parameters. Given a model, the parameters are the numbers that yield the actual distribution. In the case of a Bernoulli random variable, the single parameter was the value p. In the case of a Uniform random variable, the parameters are the a and b values that define the min and max value. Here is a list of random variables and the corresponding parameters. From now on, we are going to use the notation θ to be a vector of all the parameters:

Distribution	**Parameters**
Bernoulli(p)	$\theta = p$
Poisson(λ)	$\theta = \lambda$
Uniform(a, b)	$\theta = [a, b]$
Normal(μ, σ^2)	$\theta = [\mu, \sigma^2]$

In the real world often you don't know the "true" parameters, but you get to observe data. Next up, we will explore how we can use data to estimate the model parameters.

It turns out there isn't just one way to estimate the value of parameters. There are two main schools of thought: Maximum Likelihood Estimation (MLE) and Maximum A Posteriori (MAP). Both of these schools of thought assume that your data are independent and identically distributed (IID) samples: $X_1, X_2, \ldots X_n$.

Maximum Likelihood Estimation

Our first algorithm for estimating parameters is called Maximum Likelihood Estimation (MLE). The central idea behind MLE is to select that parameters (θ) that make the observed data the most likely.

The data that we are going to use to estimate the parameters are going to be n independent and identically distributed (IID) samples: $X_1, X_2, \ldots X_n$.

1. Likelihood

We made the assumption that our data are identically distributed. This means that they must have either the same probability mass function (if the data are discrete) or the same probability density function (if the data are continuous). To simplify our conversation about parameter estimation we are going to use the notation $f(X|\theta)$ to refer to this shared PMF or PDF. Our new notation is interesting in two ways. First, we have now included a conditional on θ which is our way of indicating that the likelihood of different values of X depends on the values of our parameters. Second, we are going to use the same symbol f for both discrete and continuous distributions.

What does likelihood mean and how is ``likelihood" different than ``probability"? In the case of discrete distributions, likelihood is a synonym for the joint probability of your data. In the case of continuous distribution, likelihood refers to the joint probability density of your data.

Since we assumed that each data point is independent, the likelihood of all of our data is the product of the likelihood of each data point. Mathematically, the likelihood of our data give parameters θ is:

$$L(\theta) = \prod_{i=1}^{n} f(X_i|\theta)$$

For different values of parameters, the likelihood of our data will be different. If we have correct parameters our data will be much more probable than if we have incorrect parameters. For that reason we write likelihood as a function of our parameters (θ).

2. Maximization

In maximum likelihood estimation (MLE) our goal is to chose values of our parameters (θ) that maximizes the likelihood function from the previous section. We are going to use the notation $\hat{\theta}$ to represent the best choice of values for our parameters. Formally, MLE assumes that:

$$\hat{\theta} = \underset{\theta}{\text{argmax}} \ L(\theta)$$

Argmax is short for Arguments of the Maxima. The argmax of a function is the value of the domain at which the function is maximized. It applies for domains of any dimension.

A cool property of argmax is that since log is a monotone function, the argmax of a function is the same as the argmax of the log of the function! That's nice because logs make the math simpler. If we find the argmax of the log of likelihood it will be equal to the armax of the likelihood. Thus for MLE we first write the Log Likelihood function (LL)

$$LL(\theta) = \log L(\theta) = \log \prod_{i=1}^{n} f(X_i|\theta) = \sum_{i=1}^{n} \log f(X_i|\theta)$$

To use a maximum likelihood estimator, first write the log likelihood of the data given your parameters. Then chose the value of parameters that maximize the log likelihood function. Argmax can be computed in many ways. All of the methods that we cover in this class require computing the first derivative of the function.

3. Bernoulli MLE Estimation

For our first example, we are going to use MLE to estimate the p parameter of a Bernoulli distribution. We are going to make our estimate based on n data points which we will refer to as IID random variables $X_1, X_2, \dots X_n$. Every one of these random variables is assumed to be a sample from the same Bernoulli, with the same p, $X_i \sim \text{Ber}(p)$. We want to find out what that p is.

Step one of MLE is to write the likelihood of a Bernoulli as a function that we can maximize. Since a Bernoulli is a discrete distribution, the likelihood is the probability mass function.

The probability mass function of a Bernoulli X can be written as $f(x) = p^x(1-p)^{1-x}$. Wow! What's up with that? It's an equation that allows us to say that the probability that $X = 1$ is p and the probability that $X = 0$ is $1 - p$. Convince yourself that when $X_i = 0$ and $X_i = 1$ the PMF returns the right probabilities. We write the PMF this way because its derivable.

Now let's do some MLE estimation:

$$
\begin{aligned}
L(\theta) &= \prod_{i=1}^{n} p^{x_i}(1-p)^{1-x_i} && \text{First write the likelihood function} \\
LL(\theta) &= \sum_{i=1}^{n} \log p^{x_i}(1-p)^{1-x_i} && \text{Then write the log likelihood function} \\
&= \sum_{i=1}^{n} x_i(\log p) + (1-x_i) log(1-p) \\
&= Y \log p + (n - Y) \log(1-p) && \text{where } Y = \sum_{i=1}^{n} x_i
\end{aligned}
$$

Great Scott! We have the log likelihood equation. Now we simply need to chose the value of p that maximizes our log-likelihood. As your calculus teacher probably taught you, one way to find the value which maximizes a function that is to find the first derivative of the function and set it equal to 0.

$$
\begin{aligned}
\frac{\delta LL(p)}{\delta p} = Y\frac{1}{p} + (n-Y)\frac{-1}{1-p} = 0 \\
\hat{p} = \frac{Y}{n} = \frac{\sum_{i=1}^{n} x_i}{n}
\end{aligned}
$$

All that work and find out that the MLE estimate is simply the sample mean...

4. Normal MLE Estimation

Practice is key. Next up we are going to try and estimate the best parameter values for a normal distribution. All we have access to are n samples from our normal which we refer to as IID random variables $X_1, X_2, \dots X_n$. We assume that for all i, $X_i \sim N(\mu = \theta_0, \sigma^2 = \theta_1)$. This example seems trickier since a normal has **two** parameters that we have to estimate. In this case θ is a vector with two values, the first is the mean (μ) parameter. The second is the variance(σ^2) parameter.

$$
\begin{aligned}
L(\theta) &= \prod_{i=1}^{n} f(X_i|\theta) \\
&= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\theta_1}} e^{-\frac{(x_i-\theta_0)^2}{2\theta_1}} && \text{Likelihood for a continuous variable is the PDF} \\
LL(\theta) &= \sum_{i=1}^{n} \log \frac{1}{\sqrt{2\pi\theta_1}} e^{-\frac{(x_i-\theta_0)^2}{2\theta_1}} && \text{We want to calculate log likelihood} \\
&= \sum_{i=1}^{n} \left[-\log(\sqrt{2\pi\theta_1}) - \frac{1}{2\theta_1}(x_i - \theta_0)^2 \right]
\end{aligned}
$$

Again, the last step of MLE is to chose values of θ that maximize the log likelihood function. In this case we can calculate the partial derivative of the LL function with respect to both θ_0 and θ_1, set both equations to equal 0 and than solve for the values of θ. Doing so results in the equations for the values $\hat{\mu} = \hat{\theta_0}$ and $\hat{\sigma^2} = \hat{\theta_1}$ that maximize likelihood. The result is: $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$ and $\hat{\sigma^2} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu})^2$.

5. Linear Transform Plus Noise

MLE is an algorithm that can be used for any probability model with a derivable likelihood function. As an example lets estimate the parameter θ in a model where there is a random variable Y such that $Y = \theta X + Z$, $Z \sim N(0, \sigma^2)$ and X is an unknown distribution.

In the case where you are told the value of X, θX is a number and $\theta X + Z$ is the sum of a gaussian and a number. This implies that $Y|X \sim N(\theta X, \sigma^2)$. Our goal is to chose a value of θ that maximizes the probability IID: $(X_1, Y_1), (X_2, Y_2), \dots (X_n, Y_n)$.

We approach this problem by first finding a function for the log likelihood of the data given θ. Then we find the value of θ that maximizes the log likelihood function. To start, use the PDF of a Normal to express the probability of $Y|X, \theta$:

$$f(Y_i|X_i, \theta) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(Y_i - \theta X_i)^2}{2\sigma^2}}$$

Now we are ready to write the likelihood function, then take its log to get the log likelihood function:

$$\begin{aligned}
L(\theta) &= \prod_{i=1}^{n} f(Y_i, X_i|\theta) && \text{Let's break up this joint} \\
&= \prod_{i=1}^{n} f(Y_i|X_i, \theta) f(X_i) && f(X_i) \text{ is independent of } \theta \\
&= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(Y_i - \theta X_i)^2}{2\sigma^2}} f(X_i) && \text{Substitute in the definition of } f(Y_i|X_i)
\end{aligned}$$

$$\begin{aligned}
LL(\theta) &= \log L(\theta) \\
&= \log \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(Y_i - \theta X_i)^2}{2\sigma^2}} f(X_i) && \text{Substitute in } L(\theta) \\
&= \sum_{i=1}^{n} \log \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(Y_i - \theta X_i)^2}{2\sigma^2}} + \sum_{i=1}^{n} \log f(X_i) && \text{Log of a product is the sum of logs} \\
&= n \log \frac{1}{\sqrt{2\pi}} - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (Y_i - \theta X_i)^2 + \sum_{i=1}^{n} \log f(X_i)
\end{aligned}$$

Remove constant multipliers and terms that don't include θ. We are left with trying to find a value of θ that maximizes:

$$\begin{aligned}
\hat{\theta} &= \underset{\theta}{\operatorname{argmax}} - \sum_{i=1}^{m} (Y_i - \theta X_i)^2 \\
&= \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{m} (Y_i - \theta X_i)^2
\end{aligned}$$

This result says that the value of θ that makes the data most likely is one that minimizes the squared error of predictions of Y. We will see in a few days that this is the basis for linear regression.

Maximum A Posteriori

MLE is great, but it is not the only way to estimate parameters! This section introduces an alternate algorithm, Maximum A Posteriori (MAP).The paradigm of MAP is that we should chose the value for our parameters that is the most likely given the data. At first blush this might seem the same as MLE, however notice that MLE chooses the value of parameters that makes the \emph{data} most likely. Formally, for IID random variables $X_1, \ldots, X_n$:

$$\theta_{\text{MAP}} = \underset{\theta}{\text{argmax}} \; f(\theta | X_1, X_2, \ldots X_n)$$

In the equation above we trying to calculate the conditional probability of unobserved random variables given observed random variables. When that is the case, think Bayes Theorem! Expand the function f using the continuous version of Bayes Theorem.

$$\begin{aligned}
\theta_{\text{MAP}} &= \underset{\theta}{\text{argmax}} \; f(\theta | X_1, X_2, \ldots X_n) && \text{Now apply Bayes Theorem} \\
&= \underset{\theta}{\text{argmax}} \; \frac{f(X_1, X_2, \ldots, X_n | \theta) g(\theta)}{h(X_1, X_2, \ldots X_n)} && \text{Ahh much better}
\end{aligned}$$

Note that f, g and h are all probability densities. I used different symbols to make it explicit that they may have different functions. Now we are going to leverage two observations. First, the data is assumed to be IID so we can decompose the density of the data given θ. Second, the denominator is a constant with respect to θ. As such its value does not affect the argmax and we can drop that term. Mathematically:

$$\begin{aligned}
\theta_{\text{MAP}} &= \underset{\theta}{\text{argmax}} \; \frac{\prod_{i=1}^{n} f(X_i | \theta) g(\theta)}{h(X_1, X_2, \ldots X_n)} && \text{Since the samples are IID} \\
&= \underset{\theta}{\text{argmax}} \; \prod_{i=1}^{n} f(X_i | \theta) g(\theta) && \text{Since } h \text{ is constant with respect to } \theta
\end{aligned}$$

As before, it will be more convenient to find the argmax of the log of the MAP function, which gives us the final form for MAP estimation of parameters.

$$\theta_{\text{MAP}} = \underset{\theta}{\text{argmax}} \; \left(\log(g(\theta)) + \sum_{i=1}^{n} \log(f(X_i | \theta)) \right)$$

Using Bayesian terminology, the MAP estimate is the mode of the "posterior" distribution for θ. If you look at this equation side by side with the MLE equation you will notice that MAP is the argmax of the exact same function \emph{plus} a term for the log of the prior.

1. Parameter Priors

In order to get ready for the world of MAP estimation, we are going to need to brush up on our distributions. We will need reasonable distributions for each of our different parameters. For example, if you are predicting a Poisson distribution, what is the right random variable type for the prior of λ?

A desiderata for prior distributions is that the resulting posterior distribution has the same functional form. We call these "conjugate" priors. In the case where you are updating your belief many times, conjugate priors makes programming in the math equations much easier.

Here is a list of different parameters and the distributions most often used for their priors:

Parameter	Distribution
Bernoulli p	Beta
Binomial p	Beta
Poisson λ	Gamma
Exponential λ	Gamma
Multinomial p_i	Dirichlet
Normal μ	Normal
Normal σ^2	Inverse Gamma

You are only expected to know the new distributions on a high level. You do not need to know Inverse Gamma. I included it for completeness.

The distributions used to represent your "prior" belief about a random variable will often have their own parameters. For example, a Beta distribution is defined using two parameters (a, b). Do we have to use parameter estimation to evaluate a and b too? No. Those parameters are called "hyperparameters". That is a term we reserve for parameters in our model that we fix before running parameter estimate. Before you run MAP you decide on the values of (a, b).

2. Dirichlet

The Dirichlet distribution generalizes Beta in same way Multinomial generalizes Bernoulli. A random variable X that is Dirichlet is parametrized as $X \sim \text{Dirichlet}(a_1, a_2, \ldots, a_m)$. The PDF of the distribution is:

$$f(X_1 = x_1, X_2 = x_2, \ldots, X_m = x_m) = K \prod_{i=1}^{m} x_i^{a_i - 1}$$

Where K is a normalizing constant.

You can intuitively understand the hyperparameters of a Dirichlet distribution: imagine you have seen $\sum_{i=1}^{m} a_i - m$ imaginary trials. In those trials you had $(a_i - 1)$ outcomes of value i. As an example consider estimating the probability of getting different numbers on a six-sided Skewed Dice (where each side is a different shape). We will estimate the probabilities of rolling each side of this dice by repeatedly rolling the dice n times. This will produce n IID samples. For the MAP paradigm, we are going to need a prior on our belief of each of the parameters $p_1 \ldots p_6$. We want to express that we lightly believe that each roll is equally likely.

Before you roll, let's imagine you had rolled the dice six times and had gotten one of each possible values. Thus the "prior" distribution would be Dirichlet$(2, 2, 2, 2, 2, 2)$. After observing $n_1 + n_2 + \cdots + n_6$ new trials with n_i results of outcome i, the "posterior" distribution is Dirichlet($2 + n_1, \ldots 2 + n_6$). Using a prior which represents one imagined observation of each outcome is called "Laplace smoothing" and it guarantees that none of your probabilities are 0 or 1.

3. Gamma

The Gamma(k, θ) distribution is the conjugate prior for the λ parameter of the Poisson distribution (It is also the conjugate for Exponential, but we won't delve into that).

The hyperparameters can be interpreted as: you saw k total imaginary events during θ imaginary time periods. After observing n events during the next t time periods the posterior distribution is Gamma($k + n, \theta + t$).

For example Gamma(10, 5) would represent having seen 10 imaginary events in 5 time periods. It is like imagining a rate of 2 with some degree of confidence. If we start with that Gamma as a prior and then see 11 events in the next 2 time periods our posterior is Gamma(21,7) which is equivalent to an updated rate of 3.

Machine Learning

Machine Learning is the subfield of computer science that gives computers the ability to perform tasks without being explicitly programmed. There are several different tasks that fall under the domain of machine learning and several different algorithms for "learning". In this chapter, we are going to focus on Classification and two classic Classification algorithms: Naive Bayes and Logistic Regression.

1. Classification

In classification tasks, your job is to use training data with feature/label pairs ($\mathbf{x}$, y) in order to estimate a function $\hat{y} = g(\mathbf{x})$. This function can then be used to make a prediction. In classification the value of y takes on one of a \textbf{discrete} number of values. As such we often chose $g(\mathbf{x}) = \underset{y}{\text{argmax}}\ \hat{P}(Y = y|\mathbf{X})$.

In the classification task you are given N training pairs: $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots, (\mathbf{x}^{(N)}, y^{(N)})$ Where $\mathbf{x}^{(i)}$ is a vector of m discrete features for the ith training example and $y^{(i)}$ is the discrete label for the ith training example.

In our introduction to machine learning, we are going to assume that all values in our training data-set are binary. While this is not a necessary assumption (both naive Bayes and logistic regression can work for non-binary data), it makes it much easier to learn the core concepts. Specifically we assume that all labels are binary $y^{(i)} \in \{0, 1\}\ \forall i$ and all features are binary $x_j^{(i)} \in \{0, 1\}\ \forall i, j$.

Naïve Bayes

Naive Bayes is a Machine Learning algorithm for the ``classification task". It make the substantial assumption (called the Naive Bayes assumption) that all features are independent of one another, given the classification label. This assumption is wrong, but allows for a fast and quick algorithm that is often useful. In order to implement Naive Bayes you will need to learn how to train your model and how to use it to make predictions, once trained.

1. Training (aka Parameter Estimation)

The objective in training is to estimate the probabilities $P(Y)$ and $P(X_i|Y)$ for all $0 < i \leq m$ features. We use the symbol $\hat{p}$ to make it clear that the probability is an estimate.

Using an MLE estimate:

$$\hat{p}(X_i = x_i|Y = y) = \frac{(\#\text{ training examples where } X_i = x_i \text{ and } Y = y)}{(\#\text{ training examples where } Y = y)}$$

Using a Laplace MAP estimate:

$$\hat{p}(X_i = x_i|Y = y) = \frac{(\#\text{ training examples where } X_i = x_i \text{ and } Y = y) + 1}{(\#\text{ training examples where } Y = y) + 2}$$

The prior probability of Y trained using an MLE estimate:

$$\hat{p}(Y = y) = \frac{(\#\text{ training examples where } Y = y)}{(\#\text{ training examples})}$$

2. Prediction

For an example with $\mathbf{x} = [x_1, x_2, \ldots, x_m]$, estimate the value of y as:

$$\hat{y} = \underset{y=\{0,1\}}{\arg\max}\ \log \hat{p}(Y = y) + \sum_{i=1}^{m} \log \hat{p}(X_i = x_i|Y = y)$$

Note that for small enough datasets you may not need to use the log version of the argmax.

3. Theory

In the world of classification when we make a prediction we want to chose the value of y that maximizes $P(Y = y|\mathbf{X})$.

$$\begin{aligned}
\hat{y} &= \underset{y=\{0,1\}}{\arg\max} P(Y = y|\mathbf{X} = \mathbf{X}) && \text{Our objective} \\
&= \underset{y=\{0,1\}}{\arg\max} \frac{P(Y = y)P(\mathbf{X} = \mathbf{x}|Y = y)}{P(\mathbf{X} = \mathbf{x})} && \text{By bayes theorem} \\
&= \underset{y=\{0,1\}}{\arg\max} P(Y = y)P(\mathbf{X} = \mathbf{x}|Y = y)) && \text{Since } P(\mathbf{X} = \mathbf{x}) \text{ is constant with respect to } Y
\end{aligned}$$

Using our training data we could interpret the joint distribution of $\mathbf{X}$ and Y as one giant multinomial with a different parameter for every combination of $\mathbf{X} = \mathbf{x}$ and $Y = y$. If for example, the input vectors are only length one. In other words $|\mathbf{x}| = 1$ and the number of values that x and y can take on are small, say binary, this is a totally reasonable approach. We could estimate the multinomial using MLE or MAP estimators and then calculate argmax over a few lookups in our table.

The bad times hit when the number of features becomes large. Recall that our multinomial needs to estimate a parameter for every unique combination of assignments to the vector $\mathbf{x}$ and the value y. If there are $|\mathbf{x}| = n$ binary features then this strategy is going to take order $\mathcal{O}(2^n)$ space and there will likely be many parameters that are estimated without any training data that matches the corresponding assignment.

Naive Bayes Assumption

The Naïve Bayes Assumption is that each feature of $\mathbf{x}$ is independent of one another given y.

The Naïve Bayes Assumption is wrong, but useful. This assumption allows us to make predictions using space and data which is linear with respect to the size of the features: $\mathcal{O}(n)$ if $|\mathbf{x}| = n$. That allows us to train and make predictions for huge feature spaces such as one which has an indicator for every word on the internet. Using this assumption the prediction algorithm can be simplified.

$$
\begin{aligned}
\hat{y} &= \underset{y=\{0,1\}}{\arg\max}\ P(Y=y)P(\mathbf{X}=\mathbf{x}|Y=y) && \text{As we last left off} \\
&= \underset{y=\{0,1\}}{\arg\max}\ P(Y=y)\prod_i P(X_i=x_i|Y=y) && \text{Naïve bayes assumption} \\
&= \underset{y=\{0,1\}}{\arg\max}\ \log P(Y=y) + \sum_i \log P(X_i=x_i|Y=y) && \text{For numerical stability}
\end{aligned}
$$

In the last step we leverage the fact that the argmax of a function is equal to the argmax of the log of a function. This algorithm is both fast and stable both when training and making predictions.

Logistic Regression

Logistic Regression is a classification algorithm (I know, terrible name. Perhaps Logistic Classification would have been better) that works by trying to learn a function that approximates $\mathrm{P}(y|x)$. It makes the central assumption that $\mathrm{P}(y|x)$ can be approximated as a sigmoid function applied to a linear combination of input features. It is particularly important to learn because logistic regression is the basic building block of artificial neural networks.

Mathematically, for a single training datapoint $(\mathbf{x}, y)$ Logistic Regression assumes:

$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma(z) \text{ where } z = \theta_0 + \sum_{i=1}^{m} \theta_i x_i$$

This assumption is often written in the equivalent forms:

$$\begin{aligned} P(Y = 1|\mathbf{X} = \mathbf{x}) &= \sigma(\theta^T \mathbf{x}) && \text{where we always set } x_0 \text{ to be 1} \\ P(Y = 0|\mathbf{X} = \mathbf{x}) &= 1 - \sigma(\theta^T \mathbf{x}) && \text{by total law of probability} \end{aligned}$$

Using these equations for probability of $Y|X$ we can create an algorithm that selects values of theta that maximize that probability for all data. I am first going to state the log probability function and partial derivatives with respect to theta. Then later we will (a) show an algorithm that can chose optimal values of theta and (b) show how the equations were derived.

An important thing to realize is that: given the best values for the parameters (θ), logistic regression often can do a great job of estimating the probability of different class labels. However, given bad , or even random, values of θ it does a poor job. The amount of ``intelligence" that you logistic regression machine learning algorithm has is dependent on having good values of θ.

1. Notation

Before we get started I want to make sure that we are all on the same page with respect to notation. In logistic regression, θ is a vector of parameters of length m and we are going to learn the values of those parameters based off of n training examples. The number of parameters should be equal to the number of features of each datapoint.

Two pieces of notation that we use often in logistic regression that you may not be familiar with are:

$$\begin{aligned} \theta^T \mathbf{x} &= \sum_{i=1}^{m} \theta_i x_i = \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_m x_m && \text{dot product, aka weighted sum} \\ \sigma(z) &= \frac{1}{1 + e^{-z}} && \text{sigmoid function} \end{aligned}$$

2. Log Likelihood

In order to chose values for the parameters of logistic regression we use Maximum Likelihood Estimation (MLE). As such we are going to have two steps: (1) write the log-likelihood function and (2) find the values of θ that maximize the log-likelihood function.

The labels that we are predicting are binary, and the output of our logistic regression function is supposed to be the probability that the label is one. This means that we can (and should) interpret the each label as a Bernoulli random variable: $Y \sim \text{Bern}(p)$ where $p = \sigma(\theta^T \mathbf{x})$.

To start, here is a super slick way of writing the probability of one datapoint (recall this is the equation form of the probability mass function of a Bernoulli):

$$P(Y = y|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})^y \cdot \left[1 - \sigma(\theta^T \mathbf{x})\right]^{(1-y)}$$

Now that we know the probability mass function, we can write the likelihood of all the data:

$$
\begin{aligned}
L(\theta) &= \prod_{i=1}^{n} P(Y = y^{(i)} | X = \mathbf{x}^{(i)}) && \text{The likelihood of independent training labels} \\
&= \prod_{i=1}^{n} \sigma(\theta^T \mathbf{x}^{(i)})^{y^{(i)}} \cdot \left[1 - \sigma(\theta^T \mathbf{x}^{(i)})\right]^{(1-y^{(i)})} && \text{Substituting the likelihood of a Bernoulli}
\end{aligned}
$$

And if you take the log of this function, you get the reported Log Likelihood for Logistic Regression. The log likelihood equation is:

$$
LL(\theta) = \sum_{i=1}^{n} y^{(i)} \log \sigma(\theta^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \log[1 - \sigma(\theta^T \mathbf{x}^{(i)})]
$$

Recall that in MLE the only remaining step is to chose parameters (θ) that maximize log likelihood.

3. Gradient of Log Likelihood

Now that we have a function for log-likelihood, we simply need to chose the values of theta that maximize it. We can find the best values of theta by using an optimization algorithm. However, in order to use an optimization algorithm, we first need to know the partial derivative of log likelihood with respect to each parameter. First I am going to give you the partial derivative (so you can see how it is used). Then I am going to show you how to derive it:

$$
\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=1}^{n} \left[y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)})\right] x_j^{(i)}
$$

4. Gradient Descent Optimization

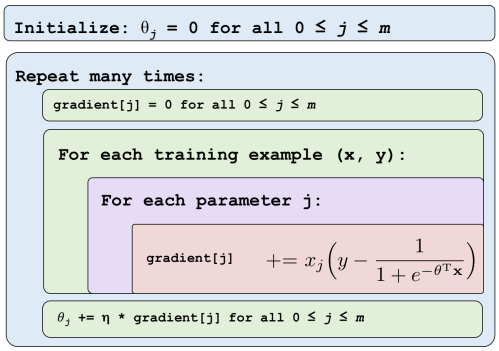
Our goal is to choosing parameters (θ) that maximize likelihood, and we know the partial derivative of log likelihood with respect to each parameter. We are ready for our optimization algorithm.

In the case of logistic regression we can't solve for θ mathematically. Instead we use a computer to chose θ. To do so we employ an algorithm called gradient descent (a classic in optimization theory). The idea behind gradient descent is that if you continuously take small steps downhill (in the direction of your negative gradient), you will eventually make it to a local minima. In our case we want to maximize our likelihood. As you can imagine, minimizing a negative of our likelihood will be equivalent to maximizing our likelihood.

The update to our parameters that results in each small step can be calculated as:

$$
\begin{aligned}
\theta_j^{\text{new}} &= \theta_j^{\text{old}} + \eta \cdot \frac{\partial LL(\theta^{\text{old}})}{\partial \theta_j^{\text{old}}} \\
&= \theta_j^{\text{old}} + \eta \cdot \sum_{i=1}^{n} \left[y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)})\right] x_j^{(i)}
\end{aligned}
$$

Where η is the magnitude of the step size that we take. If you keep updating θ using the equation above you will converge on the best values of θ. You now have an intelligent model. Here is the gradient ascent algorithm for logistic regression in pseudo-code:



Pro-tip: Don't forget that in order to learn the value of θ_0 you can simply define $\mathbf{x}_0$ to always be 1.

5. Derivations

In this section we provide the mathematical derivations for the gradient of log-likelihood. The derivations are worth knowing because these ideas are heavily used in Artificial Neural Networks.

Our goal is to calculate the derivative of the log likelihood with respect to each theta. To start, here is the definition for the derivative of a sigmoid function with respect to its inputs:

$$\frac{\partial}{\partial z}\sigma(z) = \sigma(z)[1-\sigma(z)] \qquad \text{to get the derivative with respect to } \theta, \text{ use the chain rule}$$

Take a moment and appreciate the beauty of the derivative of the sigmoid function. The reason that sigmoid has such a simple derivative stems from the natural exponent in the sigmoid denominator.

Since the likelihood function is a sum over all of the data, and in calculus the derivative of a sum is the sum of derivatives, we can focus on computing the derivative of one example. The gradient of theta is simply the sum of this term for each training datapoint.

First I am going to show you how to compute the derivative the hard way. Then we are going to look at an easier method. The derivative of gradient for one datapoint $(\mathbf{x}, y)$:

$$\begin{aligned}
\frac{\partial LL(\theta)}{\partial \theta_j} &= \frac{\partial}{\partial \theta_j} y \log \sigma(\theta^T \mathbf{x}) + \frac{\partial}{\partial \theta_j}(1-y)\log[1-\sigma(\theta^T \mathbf{x}] && \text{derivative of sum of terms} \\
&= \left[\frac{y}{\sigma(\theta^T \mathbf{x})} - \frac{1-y}{1-\sigma(\theta^T \mathbf{x})}\right]\frac{\partial}{\partial \theta_j}\sigma(\theta^T \mathbf{x}) && \text{derivative of } \log f(x) \\
&= \left[\frac{y}{\sigma(\theta^T \mathbf{x})} - \frac{1-y}{1-\sigma(\theta^T \mathbf{x})}\right]\sigma(\theta^T \mathbf{x})[1-\sigma(\theta^T \mathbf{x})]\mathbf{x}_j && \text{chain rule + derivative of sigma} \\
&= \left[\frac{y-\sigma(\theta^T \mathbf{x})}{\sigma(\theta^T \mathbf{x})[1-\sigma(\theta^T \mathbf{x})]}\right]\sigma(\theta^T \mathbf{x})[1-\sigma(\theta^T \mathbf{x})]\mathbf{x}_j && \text{algebraic manipulation} \\
&= [y-\sigma(\theta^T \mathbf{x})]\mathbf{x}_j && \text{cancelling terms}
\end{aligned}$$

6. Derivatives Without Tears

That was the hard way. Logistic regression is the building block of Artificial Neural Networks. If we want to scale up, we are going to have to get used to an easier way of calculating derivatives. For that we are going to have to welcome back our old friend the chain rule. By the chain rule:

$$\begin{aligned}
\frac{\partial LL(\theta)}{\partial \theta_j} &= \frac{\partial LL(\theta)}{\partial p} \cdot \frac{\partial p}{\partial \theta_j} && \text{Where } p = \sigma(\theta^T \mathbf{x}) \\
&= \frac{\partial LL(\theta)}{\partial p} \cdot \frac{\partial p}{\partial z} \cdot \frac{\partial z}{\partial \theta_j} && \text{Where } z = \theta^T \mathbf{x}
\end{aligned}$$

Chain rule is the decomposition mechanism of calculus. It allows us to calculate a complicated partial derivative ($\frac{\partial LL(\theta)}{\partial \theta_j}$) by breaking it down into smaller pieces.

$$\begin{aligned}
LL(\theta) &= y \log p + (1-y)\log(1-p) && \text{Where } p = \sigma(\theta^T \mathbf{x}) \\
\frac{\partial LL(\theta)}{\partial p} &= \frac{y}{p} - \frac{1-y}{1-p} && \text{By taking the derivative}
\end{aligned}$$

$$\begin{aligned}
p &= \sigma(z) && \text{Where } z = \theta^T \mathbf{x} \\
\frac{\partial p}{\partial z} &= \sigma(z)[1-\sigma(z)] && \text{By taking the derivative of the sigmoid}
\end{aligned}$$

$$\begin{aligned}
z &= \theta^T \mathbf{x} && \text{As previously defined} \\
\frac{\partial z}{\partial \theta_j} &= \mathbf{x}_j && \text{Only } \mathbf{x}_j \text{ interacts with } \theta_j
\end{aligned}$$

Each of those derivatives was much easier to calculate. Now we simply multiply them together.

$$\begin{aligned}
\frac{\partial LL(\theta)}{\partial \theta_j} &= \frac{\partial LL(\theta)}{\partial p} \cdot \frac{\partial p}{\partial z} \cdot \frac{\partial z}{\partial \theta_j} \\
&= \left[\frac{y}{p} - \frac{1-y}{1-p}\right] \cdot \sigma(z)[1-\sigma(z)] \cdot \mathbf{x}_j && \text{By substituting in for each term} \\
&= \left[\frac{y}{p} - \frac{1-y}{1-p}\right] \cdot p[1-p] \cdot \mathbf{x}_j && \text{Since } p = \sigma(z) \\
&= [y(1-p) - p(1-y)] \cdot \mathbf{x}_j && \text{Multiplying in} \\
&= [y-p]\mathbf{x}_j && \text{Expanding} \\
&= [y-\sigma(\theta^T \mathbf{x})]\mathbf{x}_j && \text{Since } p = \sigma(\theta^T \mathbf{x})
\end{aligned}$$

MLE Normal Demo

Lets manually perform maximum likelihood estimation. Your job is to chose parameter values that make the data look as likely as possible. Here are the 20 data points, which we assume come from a Normal distribution

Data = [6.3 , 5.5 , 5.4, 7.1, 4.6, 6.7, 5.3 , 4.8, 5.6, 3.4, 5.4, 3.4, 4.8, 7.9, 4.6, 7.0, 2.9, 6.4, 6.0 , 4.3]

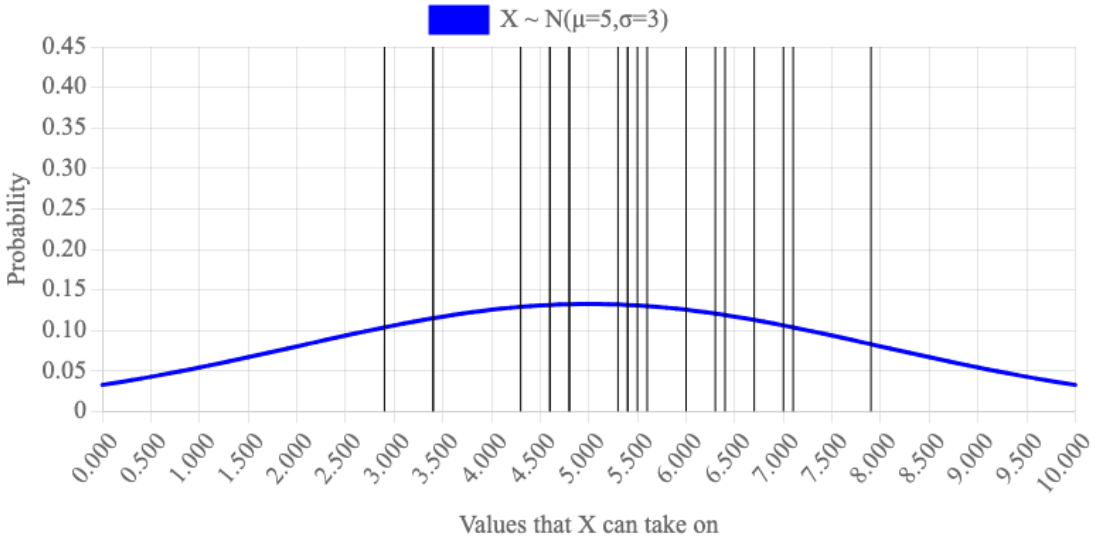
Chose your parameter estimates

Parameter μ: 5 Parameter σ: 3

Likelihood of the data given your params

Likelihood: 4.0307253523200347e-19
Log Likelihood: -399.7
Best Seen: -399.7

Your Gaussian

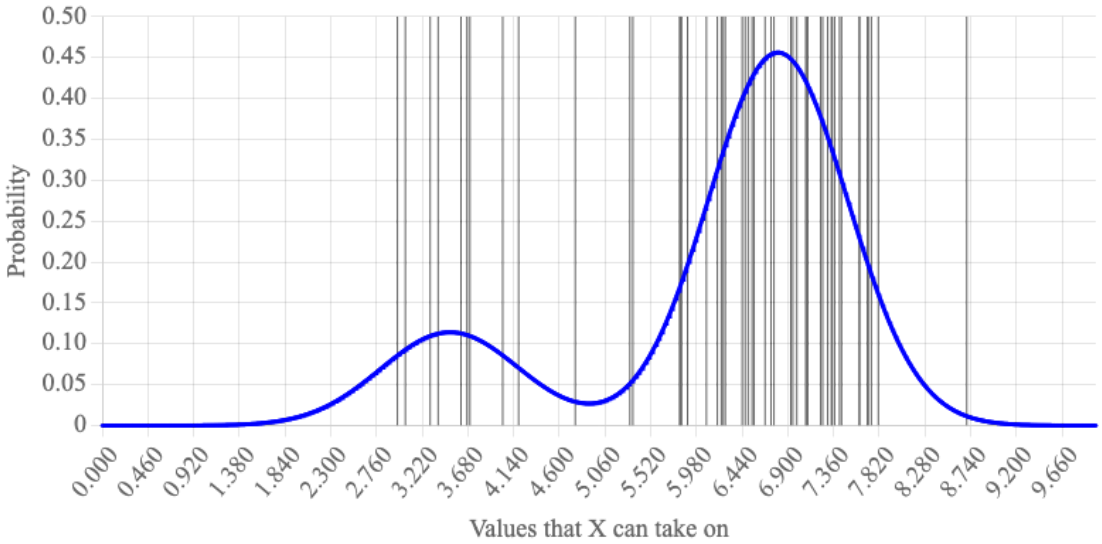


Gaussian Mixtures

Errata: This example was first written at 1:00p on Nov 10th. During class we noticed a mistake in the derivative (incorrectly assumed log of sum was sum of log). It was updated soon after class on Nov 10th.

Data = [6.47, 5.82, 8.7, 4.76, 7.62, 6.95, 7.44, 6.73, 3.38, 5.89, 7.81, 6.93, 7.23, 6.25, 5.31, 7.71, 7.42, 5.81, 4.03, 7.09, 7.1, 7.62, 7.74, 6.19, 7.3, 7.37, 6.99, 2.97, 3.3, 7.08, 6.23, 3.67, 3.05, 6.67, 6.5, 6.08, 3.7, 6.76, 6.56, 3.61, 7.25, 7.34, 6.27, 6.54, 5.83, 6.44, 5.34, 7.7, 4.19, 7.34]

Parameter t: 0.2 Parameter μ_a: 3.5 Parameter σ_a: 0.7 Parameter μ_b: 6.8 Parameter σ_b: 0.7



Likelihood: 1.847658621579746e-34
Log Likelihood: -77.7
Best Seen: -77.7

What is a Gaussian Mixture?

A Gaussian Mixture describes a random variable whose PDF could come from one of two Gaussians (or more, but we will just use two in this demo). There is a certain probability the sample will come from the first gaussian, otherwise it comes from the second. It has five parameters: 4 to describe the two gaussians and one to describe the relative weighting of the two gaussians.

Generative Code

```
from scipy import stats
def sample():
   # choose group membership
   membership = stats.bernoulli.rvs(0.2)
   if membership == 1:
      # sample from gaussian 1
      return stats.norm.rvs(3.5,0.7)
   else:
      # sample from gaussian 2
      return stats.norm.rvs(6.8,0.7)
```

Probability Density Function

$$f(X = x) = t \cdot f(A = x) + (1 - t) \cdot f(B = x)$$

st

$$A \sim N(\mu_a, \sigma_a^2)$$

$$B \sim N(\mu_b, \sigma_b^2)$$

Putting it all together, the PDF of a Gaussian Mixture is:

$$f(x) = t \cdot \left(\frac{1}{\sqrt{2\pi}\sigma_a} e^{-\frac{1}{2}(\frac{x-\mu_a}{\sigma_a})^2}\right) + (1-t) \cdot \left(\frac{1}{\sqrt{2\pi}\sigma_b} e^{-\frac{1}{2}(\frac{x-\mu_b}{\sigma_b})^2}\right)$$

MLE for Gaussian Mixture

Special note: even though the generative story has a bernoulli (group membership) it is never observed. MLE maximizes the likelihood of the observed data.

Let $\vec{\theta} = [t, \mu_a, \mu_b, \sigma_a, \sigma_b]$ be the parameters. Because the math will get long I will use θ as notation in place of $\vec{\theta}$. Just keep in mind that it is a vector.

The MLE idea is to chose values of θ which maximize log likelihood. All optimization methods require us to calculate the partial derivatives of the thing we want to optimize (log likelihood) with respect to the values we can change (our parameters).

Likelihood function

$$\begin{aligned} L(\theta) &= \prod_i^n f(x_i|\theta) \\ &= \prod_i^n \left[t \cdot \left(\frac{1}{\sqrt{2\pi}\sigma_a} e^{-\frac{1}{2}(\frac{x_i-\mu_a}{\sigma_a})^2}\right) + (1-t) \cdot \left(\frac{1}{\sqrt{2\pi}\sigma_b} e^{-\frac{1}{2}(\frac{x_i-\mu_b}{\sigma_b})^2}\right)\right] \end{aligned}$$

Log Likelihood function

$$\begin{aligned} LL(\theta) &= \log L(\theta) \\ &= \log \prod_i^n f(x_i|\theta) \\ &= \sum_i^n \log f(x_i|\theta) \end{aligned}$$

That is sufficient for now, but if you wanted to expand out the term you would get:

$$LL(\theta) = \sum_i^n \log \left[t \cdot \left(\frac{1}{\sqrt{2\pi}\sigma_a} e^{-\frac{1}{2}(\frac{x_i-\mu_a}{\sigma_a})^2}\right) + (1-t) \cdot \left(\frac{1}{\sqrt{2\pi}\sigma_b} e^{-\frac{1}{2}(\frac{x_i-\mu_b}{\sigma_b})^2}\right)\right]$$

Derivative of LL with respect to θ

Here is an example of calculating a partial derivative with respect to one of the parameters, μ_a. You would need a derivative like this for all parameters.

Caution: When I first wrote this demo I thought it would be a simple derivative . It is not so simple because the log has a sum in it. As such the log term doesn't reduce. The log still serves to make the outer $\prod$ into a $\sum$. As such the LL partial derivatives are solvable, but the proof uses quite a lot of chain rule.

Takeaway: The main takeaway from this section (in case you want to skip the derivative proof) is that the resulting derivative is complex enough that we will want a way to compute argmax without having to set that derivative equal to zero and solving for μ_a. Enter gradient descent!

A good first step when doing a huge derivative of a log likelihood function is to think of the derivative for the log of likelihood of a single datapoint. This is the inner sum in the log likelihood expression:

$$\frac{d}{d\mu_a} \log f(x_i|\theta)$$

Before we start: notice that μ_a does not show up in this term from $f(x_i|\theta)$:

$$(1-t) \cdot \left(\frac{1}{\sqrt{2\pi}\sigma_b} e^{-\frac{1}{2}(\frac{x_i-\mu_b}{\sigma_b})^2}\right) = K$$

In the proof, when we encounter this term, we are going to think of it as a constant which we call K. Ok, lets go for it!

$$
\begin{aligned}
&\frac{d}{d\mu_a}\log f(x_i|\theta) \\
&\quad = \frac{1}{f(x_i|\theta)}\frac{d}{d\mu_a}f(x_i|\theta) && \text{chain rule on } \log \\
&\quad = \frac{1}{f(x_i|\theta)}\frac{d}{d\mu_a}\left[t\cdot\left(\frac{1}{\sqrt{2\pi}\sigma_a}e^{-\frac{1}{2}(\frac{x_i-\mu_a}{\sigma_a})^2}\right)+K\right] && \text{substitute in } f(x_i|\theta) \\
&\quad = \frac{1}{f(x_i|\theta)}\frac{d}{d\mu_a}\left[t\cdot\left(\frac{1}{\sqrt{2\pi}\sigma_a}e^{-\frac{1}{2}(\frac{x_i-\mu_a}{\sigma_a})^2}\right)\right] && \frac{d}{d\mu_a}K=0 \\
&\quad = \frac{t}{f(x_i|\theta)\sqrt{2\pi}\sigma_a}\cdot\frac{d}{d\mu_a}e^{-\frac{1}{2}(\frac{x_i-\mu_a}{\sigma_a})^2} && \text{pull out const} \\
&\quad = \frac{t}{f(x_i|\theta)\sqrt{2\pi}\sigma_a}\cdot e^{-\frac{1}{2}(\frac{x_i-\mu_a}{\sigma_a})^2}\cdot\frac{d}{d\mu_a}-\frac{1}{2}(\frac{x_i-\mu_a}{\sigma_a})^2 && \text{chain on } e^x \\
&\quad = \frac{t}{f(x_i|\theta)\sqrt{2\pi}\sigma_a}\cdot e^{-\frac{1}{2}(\frac{x_i-\mu_a}{\sigma_a})^2}\cdot\left[-(\frac{x_i-\mu_a}{\sigma_a})\frac{d}{d\mu_a}(\frac{x_i-\mu_a}{\sigma_a})\right] && \text{chain on } x^2 \\
&\quad = \frac{t}{f(x_i|\theta)\sqrt{2\pi}\sigma_a}\cdot e^{-\frac{1}{2}(\frac{x_i-\mu_a}{\sigma_a})^2}\cdot\left[-(\frac{x_i-\mu_a}{\sigma_a})\cdot\frac{-1}{\sigma_a}\right] && \text{final derivative} \\
&\quad = \frac{t}{f(x_i|\theta)\sqrt{2\pi}\sigma_a^3}\cdot e^{-\frac{1}{2}(\frac{x_i-\mu_a}{\sigma_a})^2}\cdot(x_i-\mu_a) && \text{simplify}
\end{aligned}
$$

That was for a single data-point. For the full dataset:

$$
\begin{aligned}
\frac{dLL(\theta)}{d\mu_a} &= \sum_i^n \frac{d}{d\mu_a}\log f(x_i|\theta) \\
&= \sum_i^n \frac{t}{f(x_i|\theta)\sqrt{2\pi}\sigma_a^3}\cdot e^{-\frac{1}{2}(\frac{x_i-\mu_a}{\sigma_a})^2}\cdot(x_i-\mu_a)
\end{aligned}
$$

This process should be repeated for all five parameters! Now, how should we find a value of μ_a, which, in the presence of the other settings to parameters, and the data, makes this derivative zero? Setting the derivative = 0 and solving for μ_a is not going to work.

Use an Optimizer to Estimate Params

Once we have a LL function and the derivative of LL with respect to each parameter we are ready to compute argmax using an optimizer. In this case the best choice would probably be gradient ascent (or gradient descent with negative log likelihood).

$$
\nabla_\theta LL(\theta) = \begin{bmatrix}
\frac{dLL(\theta)}{dt} \\
\frac{dLL(\theta)}{d\mu_a} \\
\frac{dLL(\theta)}{d\mu_b} \\
\frac{dLL(\theta)}{d\sigma_a} \\
\frac{dLL(\theta)}{d\sigma_b}
\end{bmatrix}
$$

Probability and Babies

What is the probability that Laura gives birth today (given that she hasn't given birth up until today)?

Today's Date	17/0/2022

Due Date	18/1/2021

Probability of delivery today: **0.000**
Probability of delivery in next 7 days: **0.000**
Current days past due date: **365** days
Unconditioned probability mass before today: **1.000**

How likely is delivery, in humans, relative to the due date? There have been millions of births which gives us a relatively good picture [1]. The length of human pregnancy varies by quite a lot! Have you heard that it is 9 months? That is a rough, point estimate. The mean duration of pregnancy is 278.6 days, and pregnancy length has a standard deviation (SD) of 12.5 days. This distribution is not normal, but roughly matches a "skewed normal". This is a general probability mass function for the first pregnancy collected from hundreds of thousands of women (this PMF is very similar across demographics, but changes based on whether the woman has given birth before):

Of course, we have more information. Specifically, we know that Laura **hasn't** given birth up until today (we will update this example when that changes). We also know that babies which are over 14 days late are "induced" on day 14. How likely is delivery given that we haven't delivered up until today? Note that the y-axis is scalled differently:

I'm Curious

Implementation notes: this calculation was performed by storing the PDF as a list of (day, probability) points. These values are sometimes called weighted samples, or "particles" and are the key component to a "particle filtering" approach. After we observe no-delivery, we set the probability of every point which has a day before today to be 0, and then re-normalize the remaining points (aka we "filter" the "particles"). This is convenient because the "posterior" belief doesn't follow a simple equation -- using particles means we never have to write that equation down in our code.

Three friends have the exact same due date (Really! this isn't a hypothetical) What is the probability that all three couples deliver on the exact same day?

Probability of three couples on the same day: **-1.4111754409071847e+44**

How did we get that number? Let p_i be the probability that one baby is delivered on day i -- this number can be read off the probability mass function. Let X_i be the event that all three babies are delivered on day i. Note that the event X_i is mutually exclusive with the event that all three babies are born on another day (So for example, X_1 is mutually exclusive with X_2, X_3 etc). Let $N = 3$ be the event that all babies are born on the same day:

$$\begin{aligned} \Pr(N = 3) &= \sum_i \Pr(X_i) && \text{Since days are mutually exclusive} \\ &= \sum_i p_i^3 && \text{Since the three couples are independent} \end{aligned}$$

---

[1] Predicting delivery date by ultrasound and last menstrual period in early gestation

Acknowledgements: This problem was first posed to me by Chris Gregg.