CS 70 Discrete Mathematics and Probability Theory
Spring 2016 Rao and Walrand Note 7

1 Bijections

The notion of a mathematical *function*, i.e. a mapping f from an input set A to an output set B, is ubiquitous in our everyday lives. For example, your professor might assign you a letter grade of A through F based on a function which maps your numerical grade, i.e. an integer from 0 to 100, to the set $\{\text{“}A\text{”}, \text{“}B\text{”}, \text{“}C\text{”}, \text{“}D\text{”}, \text{“}F\text{”}\}$. Or consider the process of paying federal taxes, in which your income level in dollars is mapped via a function to a tax bracket which dictates the percentage of tax you must pay. Both of these are examples of functions, which most generally can be described using the notation $f : A \mapsto B$. Here, A is a set called the *domain* of f, and B is a set known as the *range* of f. Of all the possible functions in the wild, there is a special type which turns out to be particularly useful in computer science, specifically in the study of cryptography — namely, the notion of a *bijective* function. It is this class of functions which we study in this lecture.

Sanity check! Consider the “birthday function” f, which given a person’s birth date, outputs the age of that person. What are the domain A and range B of f?

Intuitively, a *bijection* is a function with the property that any output of the function can be uniquely mapped back to some input. More formally, a function $f : A \mapsto B$ is a bijection iff for all $b \in B$, there exists a unique *pre-image* $a \in A$ such that $f(a) = b$. Let us demonstrate with the following example of a function $f : \{0, \ldots, m-1\} \mapsto \{0, \ldots, m-1\}$:

$$f(x) = x + 1 \bmod m.$$

Here, f is a bijection since the unique pre-image of any $y \in \{0, \ldots, m-1\}$ is $y - 1$. The special case of $m = 4$ is depicted in Figure 1; note the special property that each element on the right side has precisely one matching element on the left side.

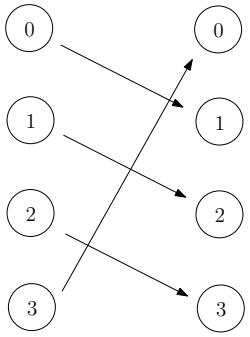


Figure 1: The bijection $f(x) = x + 1 \bmod 4$ with domain $A = \{0, \ldots, 3\}$ and range $B = \{0, \ldots, 3\}$.

Sanity check! Consider the same function $f(x) = x + 1 \bmod 4$, except now with $A = \{0, \ldots, 4\}$ and range $B = \{0, \ldots, 3\}$. Is f still a bijection? Why not? (Give two elements of A which map to the same element of B.)

If you stare at Figure 1 long enough, you might realize that the property of being a bijection is actually equivalent to having two separate properties: (1) Each element on the right has *some* pre-image on the left, and (2) if two arrows are incident on an element on the right, then both arrows must originate from the same element on the left. More formally, these properties can be stated as:

1. f is *onto* or *surjective*: Any $b \in B$ has a pre-image in A, i.e. for all $b \in B$ there exists an $a \in A$ such that $f(a) = b$.
2. f is *one-to-one* or *injective*: For all $a, a' \in A$, if $f(a) = f(a')$ then $a = a'$.

To help solidify our understanding of bijections, let us consider two more functions. First, recall the example given at the start of this lecture about the function mapping a numerical grade in the set $\{0, 1, \ldots, 100\}$ to a letter grade $\{$"*A*", "*B*", "*C*", "*D*", "*F*"$\}$. Is this a bijection? *No*, since each letter grade has many numerical grades mapped to it. (In fact, it is worth noting that there *cannot* exist a bijection between these two sets, since they have different sizes. This connection between bijections and set sizes is the key to Cantor's celebrated idea that there is more than one notion of infinity!) Next, consider the following function g mapping $\{0, \ldots, m-1\}$ to itself:

$$g(x) = 2x \bmod m. \tag{1}$$

It turns out that g is only a bijection if m is odd.

Sanity check!

1. Show that for the value $m = 4$, the function g is not a bijection since it is not one-to-one. Can you generalize your proof to arbitrary m?
2. Draw a diagram analogous to Figure 1 to show that for $m = 5$, the function g is a bijection.

A nice alternate way of thinking about bijectivity is the following, which will prove useful in our discussion on cryptography in this lecture.

Lemma 7.1. *A function $f : A \to A$ is a bijection iff there is an inverse function $g : A \to A$ such that $g(f(x)) = x$ and $f(g(y)) = y$ for all $x, y \in A$.*

Proof. We give a direct proof. First assume there exists an inverse function g as described in the claim. Then, to see that f is one-to-one, note that whenever $f(x) = f(x')$, we have that $x = g(f(x)) = g(f(x')) = x'$, as desired. To see that f is onto, note that for any $y \in A$ we have $f(g(y)) = y$, so $g(y) \in A$ is the pre-image of y.

Conversely, assume f is bijective. This means every $y \in A$ has a unique pre-image $x \in A$ such that $f(x) = y$. Define $g \colon A \to A$ by $g(y) = x$. Then we see that by construction, we have $f(g(y)) = y$, and similarly, $g(f(x)) = g(y) = x$. This shows that g is the inverse function of f, as desired. □

Sanity check! Show that the function g from Equation (2) is a bijection for the case $m = 5$ by explicitly describing an inverse function as in Lemma 7.1.

2 Fermat's Little Theorem

In 1640, Pierre de Fermat stated one of the fundamental results of elementary number theory, nowadays known as *Fermat's Little Theorem*. Ultimately, our goal in this lecture is to study the use of bijections in cryptography, and it turns out Fermat's Little Theorem will be a crucial tool in this venture. Thus, we state and prove it now. Its proof also uses the concept of a bijection.

Theorem 7.1. [Fermat's Little Theorem] *For any prime p and any $a \in \{1,2,\ldots,p-1\}$, we have $a^{p-1} \equiv 1 \bmod p$.*

Proof. Let $S = \{1,2,\ldots,p-1\}$. We give a direct proof consisting of two parts. First, we show that the function $f : S \to S$ such that $f(x) \equiv ax \bmod p$ is a bijection. Second, we use the fact that f is a bijection to prove the claim.

Sanity check! Verify that for $a = 3, p = 7$, the function $f(x) \equiv ax \bmod p$ is a bijection corresponding to the image below.

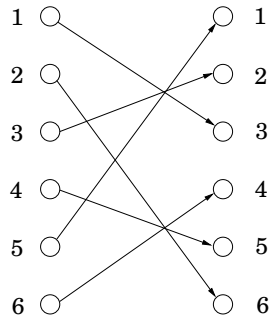


Figure 2: Multiplication by (3 mod 7)

To prove that f is indeed a bijection, since the domain and range of f are the same set S, it suffices to prove that $ax \bmod p$ and $ax' \bmod p$ are distinct if $x \neq x'$. To prove the latter, assume for sake of contradiction that there exist distinct $x, x' \in S$ such that

$$a \cdot x \equiv a \cdot x' \pmod{p}.$$

Then, since by definition a is non-zero, and since a and p are co-prime, then by Theorem 6.2, we know a has a multiplicative inverse a^{-1} modulo p. Multiplying both sides of the equation above by a^{-1} thus yields $x \equiv x' \pmod{p}$, which contradicts our assumption that x and x' are distinct. We conclude that f is a bijection.

Let us now prove our main claim. Since f is a bijection, its image is S; this implies that the product of all elements in S equals the product of all elements in the image of f. Specifically, we have

$$(p-1)! \equiv a^{p-1} \cdot (p-1)! \pmod{p}.$$

Since $(p-1)!$ is relatively prime to p, it again follows by Theorem 6.2 that we can divide both sides of this equation by $(p-1)!$, yielding the desired statement. □

3 Public Key Encryption and RSA

One of the oldest and most fundamental problems facing mankind has been: How to send a message securely from a sender, say Alice, to a receiver Bob, so that an eavesdropper Eve gains little to no information about the message? As you might imagine, this task has a host of applications in areas ranging from banking to the military. In fact, Julius Caesar himself was known to use an encryption scheme nowadays known as the *Caesar cipher* in order to protect messages of military significance. Unfortunately, Caesar's cipher had a significant drawback: In order to encode and decode messages, one needed to know a *secret key*. This led to a chicken-and-the-egg scenario — how does Alice share her secret key with Bob without Eve learning anything about it to begin with? In the 1970s, a breakthrough solution to this problem was discovered in the form of *public key encryption*.

In public key encryption, there are *two* keys: One for encryption, E, which is public knowledge to everyone, and one for decryption, D, which is known only by the receiver, Bob. In this way, *anyone in the world* can send Bob a secret message: The sender encodes the message using E, sends the encrypted message to Bob, who then decrypts it using D. How could such a scheme be possible? It turns out the secret ingredient is a special bijection which is easy to compute (i.e. anyone in the world can do it), but believed to be very difficult to *invert* without knowledge of a secret key, which only Bob possesses. This special bijection is known as the RSA function, named after its inventors Ronald Rivest, Adi Shamir and Leonard Adleman, and is as follows:

$$E(x) \equiv x^e \bmod N,$$

where $N = pq$ for two large primes p and q, $E : \{0,\ldots,N-1\} \mapsto \{0,\ldots,N-1\}$, and e is relatively prime to $(p-1)(q-1)$. It may not be clear *a priori* that this is indeed a bijection; we shall prove this shortly. The inverse of the RSA function is given by the decryption operation:

$$D(x) \equiv x^d \bmod N$$

where d is the multiplicative inverse of e mod $(p-1)(q-1)$. RSA now works as follows: The public key is (N,e), known to everyone in the world, and the private key is d, known only to Bob. To send a secret message $x \in \{1,\ldots,N-1\}$ to Bob, Alice applies the *encryption function* E to x to obtain a *ciphertext* $E(x)$, which she then sends to Bob. Upon receipt of Alice's ciphertext $E(x)$, Bob applies his *decryption function* D to recover x. This should convince you that RSA allows you to indeed encrypt and decrypt a message x. But is it *secure*? We defer this discussion to Section 3.1. For now, let us prove that indeed $D(E(x)) = x$ for all $x \in \{1,\ldots,N-1\}$, which in turn (by Theorem 7.1) implies that E is a bijection.

Theorem 7.2. *For E and D as defined above, we have $D(E(x)) = x \bmod N$ for all $x \in \{0,1,\ldots,N-1\}$.*

Proof. To prove the statement, we have to show that

$$(x^e)^d = x \bmod N \qquad \text{for every } x \in \{0,1,\ldots,N-1\}, \tag{2}$$

where recall $N = pq$ for primes p and q, $\gcd(e,(p-1)(q-1)) = 1$, and d is the multiplicative inverse of e mod $(p-1)(q-1)$. Let's consider the exponent, which is ed. By definition of d, we know that $ed = 1 \bmod (p-1)(q-1)$; hence we can write $ed = 1 + k(p-1)(q-1)$ for some integer k, and therefore

$$x^{ed} - x = x^{1+k(p-1)(q-1)} - x = x(x^{k(p-1)(q-1)} - 1). \tag{3}$$

Looking back at Equation (2), our goal is to show that this last expression in Equation (3) is equal to 0 mod N for every x. To do this, we show that the expression is divisible both by p and q; thus, since p and q are primes, the expression must also be divisible by $N = pq$, which will yield our claim.

We begin by showing that $x(x^{k(p-1)(q-1)} - 1)$ in (3) is divisible by p. We have two cases to consider:

Case 1: (x *is a multiple of* p) In this case, p clearly divides the expression in (3) since $p \mid x$.

Case 2: (x *is not a multiple of* p) Since $x \neq 0 \bmod p$, we can use Fermat's Little Theorem to deduce that $x^{p-1} = 1 \bmod p$. Then $(x^{p-1})^{k(q-1)} \equiv 1^{k(q-1)} \bmod p$, which implies that $x^{k(p-1)(q-1)} - 1 = 0 \bmod p$, and so p divides the expression in (3).

The proof that $q \mid x(x^{k(p-1)(q-1)} - 1)$ is identical. This completes the proof. □

3.1 Security of RSA

We have thus far shown that the RSA protocol is *correct*, in the sense that Alice can encrypt messages in such a way that Bob can reliably decrypt them again. But how do we know that it is *secure*, i.e., that Eve cannot obtain any information about Alice's message x from the ciphertext $E(x)$? To be clear, there is no known formal proof of this fact. Rather, the security of RSA hinges upon the following widely held *assumption*:

Given N, e and $y = x^e \bmod N$, there is no efficient algorithm for determining x.

To help us appreciate why this assumption may be true, let us brainstorm how Eve might try to guess x:

- (Brute force) The naive approach would be via brute force — simply try all possible values of x, each time checking whether $x^e = y \bmod N$. But this approach is unbelievably inefficient; Eve would have to try $O(N)$ values of x, which if N is a 512-bit number, boils down to 2^{512} values of x to try — this number is larger than estimates for the age of the Universe in femtoseconds!
- (Reduction to factoring) Eve could be more clever and instead attempt to factor N to into its prime factors p and q; then, she could compute d by determining the multiplicative inverse of $e \bmod (p-1)(q-1)$. But this requires the ability to factor large numbers, a task which is *also* considered impossible to solve efficiently.
- (Computing $(p-1)(q-1)$ directly) Finally, Eve could try to compute $(p-1)(q-1)$ without factoring N; but it turns out this is equivalent to factoring N.

Thus, the hardness of RSA depends on the presumed difficulty of factoring large numbers, known as the *Factoring Problem*. In fact, the Factoring Problem occupies a special place in theoretical computer science. It is one of the few known problems which is neither known to be efficiently solvable (i.e. in the complexity class P), nor intractable (i.e. complete for the class NP). However, there *has* been a remarkable development over the last two decades; it turns out the Factoring Problem can be solved efficiently on a *quantum* computer!

Sanity check! Our discussion regarding the brute force approach above deserves a moment's reflection, in particular with respect to the following common fallacy: Namely, in order to try all possible factors x of N requires $O(N)$ iterations, which is often interpreted as "polynomial time" since the algorithm runs in time

linear in N. Why is this algorithm *not* "polynomial time"? (Hint: What is the relevant quantity with respect to which we typically measure complexity?)

3.2 Implementations of RSA

We close this note with a brief discussion of implementation issues regarding RSA. Since we have argued that breaking RSA is impossible because *factoring* would take a very long time, we should check that the computations that Alice and Bob themselves have to perform are much simpler, and can be done efficiently.

There are really only two non-trivial things that Alice and Bob have to do:

1. Bob has to find prime numbers p and q, each having many (say, 512) bits.
2. Both Alice and Bob have to compute exponentials mod N. (Alice has to compute $x^e \bmod N$, and Bob has to compute $y^d \bmod N$.)

Both of these tasks can be carried out efficiently. The first requires the implementation of an efficient test for primality as well as a rich source of primes. You will learn how to tackle each of these tasks in the algorithms course CS170. The second requires an efficient algorithm for modular exponentiation known as "repeated squaring", which is not very difficult, but will also be discussed in detail in CS170.

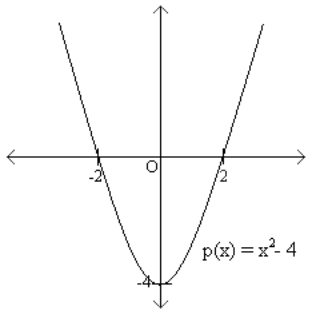
To summarize, then, in the RSA protocol Bob need only perform simple calculations such as multiplication, exponentiation and primality testing to determine the encryption and decryption keys. Similarly, Alice and Bob need only perform simple calculations to lock and unlock the secret message respectively — operations that any pocket computing device could handle. By contrast, to unlock the message without the key, Eve would have to perform operations like factoring large numbers, which (at least according to widely accepted belief) requires more computational power than all of the world's most sophisticated computers combined! This compelling guarantee of security without the need for private keys explains why the RSA cryptosystem is such a revolutionary development in cryptography.

Polynomials

Polynomials constitute a rich class of functions which are both easy to describe and widely applicable in topics ranging from Fourier analysis to computational geometry. In this note, we will discuss properties of polynomials which make them so useful. We will then describe how to take advantage of these properties to develop a secret sharing scheme.

Recall from your high school math that a *polynomial* in a single variable is of the form $p(x) = a_d x^d + a_{d-1}x^{d-1} + \ldots + a_0$. Here the *variable* x and the *coefficients* a_i are usually real numbers. For example, $p(x) = 5x^3 + 2x + 1$, is a polynomial of *degree* $d = 3$. Its coefficients are $a_3 = 5$, $a_2 = 0$, $a_1 = 2$, and $a_0 = 1$. Polynomials have some remarkably simple, elegant and powerful properties, which we will explore in this note.

First, a definition: we say that a is a *root* of the polynomial $p(x)$ if $p(a) = 0$. For example, the degree 2 polynomial $p(x) = x^2 - 4$ has two roots, namely 2 and -2, since $p(2) = p(-2) = 0$. If we plot the polynomial $p(x)$ in the x-y plane, then the roots of the polynomial are just the places where the curve crosses the x axis:

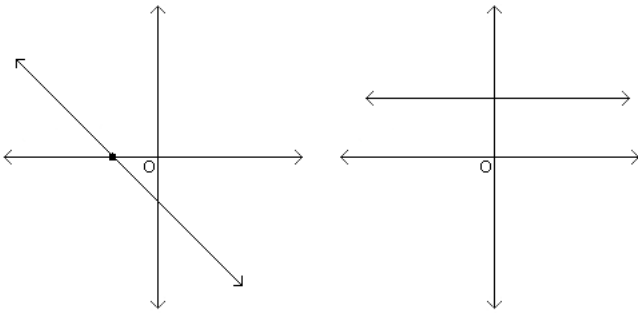


We now state two fundamental properties of polynomials that we will prove in due course.

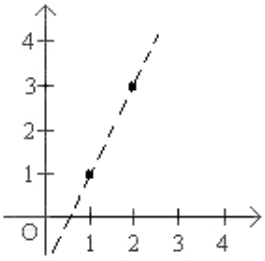
Property 1: A non-zero polynomial of degree d has at most d roots.

Property 2: Given $d+1$ pairs $(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$, with all the x_i distinct, there is a unique polynomial $p(x)$ of degree (at most) d such that $p(x_i) = y_i$ for $1 \leq i \leq d+1$.

Let us consider what these two properties say in the case that $d = 1$. A graph of a linear (degree 1) polynomial $y = a_1 x + a_0$ is a line. Property 1 says that if a line is not the x-axis (i.e. if the polynomial is not $y = 0$), then it can intersect the x-axis in at most one point.



Property 2 says that two points uniquely determine a line.



Polynomial Interpolation

Property 2 says that two points uniquely determine a degree 1 polynomial (a line), three points uniquely determine a degree 2 polynomial, four points uniquely determine a degree 3 polynomial, and so on. Given $d+1$ pairs $(x_1,y_1),\ldots,(x_{d+1},y_{d+1})$, how do we determine the polynomial $p(x) = a_d x^d + \ldots + a_1 x + a_0$ such that $p(x_i) = y_i$ for $i = 1$ to $d+1$? We will give an efficient algorithms for reconstructing the coefficients $a_0,\ldots,a_d$, and therefore the polynomial $p(x)$.

The method is called *Lagrange interpolation*: Let us start by solving an easier problem. Suppose that we are told that $y_1 = 1$ and $y_j = 0$ for $2 \le j \le d+1$. Now can we reconstruct $p(x)$? Yes, this is easy! Consider $q(x) = (x-x_2)(x-x_3)\cdots(x-x_{d+1})$. This is a polynomial of degree d (the x_i's are constants, and x appears d times). Also, we clearly have $q(x_j) = 0$ for $2 \le j \le d+1$. But what is $q(x_1)$? Well, $q(x_1) = (x_1-x_2)(x_1-x_3)\cdots(x_1-x_{d+1})$, which is some constant not equal to 0. Thus if we let $p(x) = q(x)/q(x_1)$ (dividing is ok since $q(x_1) \neq 0$), we have the polynomial we are looking for. For example, suppose you were given the pairs $(1,1)$, $(2,0)$, and $(3,0)$. Then we can construct the degree $d = 2$ polynomial $p(x)$ by letting $q(x) = (x-2)(x-3) = x^2 - 5x + 6$, and $q(x_1) = q(1) = 2$. Thus, we can now construct $p(x) = q(x)/q(x_1) = (x^2 - 5x + 6)/2$.

Of course the problem is no harder if we single out some arbitrary index i instead of 1: i.e. $y_i = 1$ and $y_j = 0$ for $j \neq i$. Let us introduce some notation: let us denote by $\Delta_i(x)$ the degree d polynomial that goes through these $d+1$ points. Then $\Delta_i(x) = \frac{\Pi j \neq i (x-x_j)}{\Pi j \neq i (x_i - x_j)}$.

Let us now return to the original problem. Given $d+1$ pairs $(x_1,y_1),\ldots,(x_{d+1},y_{d+1})$, we first construct the $d+1$ polynomials $\Delta_1(x),\ldots,\Delta_{d+1}(x)$. Now we can write $p(x) = \sum_{i=1}^{d+1} y_i \Delta_i(x)$. Why does this work? First notice that $p(x)$ is a polynomial of degree d as required, since it is the sum of polynomials of degree d. And when it is evaluated at x_i, d of the $d+1$ terms in the sum evaluate to 0 and the *i-th* term evaluates to y_i times 1, as required.

As an example, suppose we want to find the degree-2 polynomial $p(x)$ that passes through the three points

$(1,1)$, $(2,2)$ and $(3,4)$. The three polynomials Δ_i are as follows: If $d = 2$, and $x_i = i$, for instance, then

$$\Delta_1(x) = \frac{(x-2)(x-3)}{(1-2)(1-3)} = \frac{(x-2)(x-3)}{2} = \frac{1}{2}x^2 - \frac{5}{2}x + 3;$$
$$\Delta_2(x) = \frac{(x-1)(x-3)}{(2-1)(2-3)} = \frac{(x-1)(x-3)}{-1} = -x^2 + 4x - 3;$$
$$\Delta_3(x) = \frac{(x-1)(x-2)}{(3-1)(3-2)} = \frac{(x-1)(x-2)}{2} = \frac{1}{2}x^2 - \frac{3}{2}x + 1.$$

The polynomial $p(x)$ is therefore given by

$$p(x) = 1 \cdot \Delta_1(x) + 2 \cdot \Delta_2(x) + 4 \cdot \Delta_3(x) = \frac{1}{2}x^2 - \frac{1}{2}x + 1.$$

You should verify that this polynomial does indeed pass through the above three points.

Proof of Property 2

We would like to prove property 2:

Property 2: Given $d+1$ pairs $(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$, with all the x_i distinct, there is a unique polynomial $p(x)$ of degree (at most) d such that $p(x_i) = y_i$ for $1 \leq i \leq d+1$.

We have shown how to find a polynomial $p(x)$ such that $p(x_i) = y_i$ for $d+1$ pairs $(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$. This proves part of property 2 (the existence of the polynomial). How do we prove the second part, that the polynomial is unique? Suppose for contradiction that there is another polynomial $q(x)$ such that $q(x_i) = y_i$ for all $d+1$ pairs above. Now consider the polynomial $r(x) = p(x) - q(x)$. Since we are assuming that $q(x)$ and $p(x)$ are different polynomials, $r(x)$ must be a non-zero polynomial of degree at most d. Therefore, property 1 implies it can have at most d roots. But on the other hand $r(x_i) = p(x_i) - q(x_i) = 0$ on $d+1$ distinct points. Contradiction. Therefore, $p(x)$ is the unique polynomial that satisfies the $d+1$ conditions.

Polynomial Division

Let's take a short digression to discuss polynomial division, which will be useful in the proof of property 1. If we have a polynomial $p(x)$ of degree d, we can divide by a polynomial $q(x)$ of degree $\leq d$ by using long division. The result will be:

$$p(x) = q(x)q'(x) + r(x)$$

where $q'(x)$ is the quotient and $r(x)$ is the remainder. The degree of $r(x)$ must be smaller than the degree of $p(x)$.

Example. We wish to divide $p(x) = x^3 + x^2 - 1$ by $q(x) = x - 1$:

$$\begin{array}{r|l} & X^2 + 2X + 2 \\ \hline X-1 & X^3 + X^2 \quad\quad - 1 \\ & \underline{-X^3 + X^2} \\ & \quad 2X^2 \\ & \quad \underline{-2X^2 + 2X} \\ & \quad\quad 2X - 1 \\ & \quad\quad \underline{-2X + 2} \\ & \quad\quad\quad 1 \end{array}$$

Now $p(x) = x^3 + x^2 - 1 = (x-1)(x^2 + 2x + 2) + 1$, $r(x) = 1$ and $q'(x) = x^2 + 2x + 2$.

Proof of Property 1

Now let us turn to property 1: a non-zero polynomial of degree d has at most d roots.The idea of the proof is as follows. We will prove the following claims:

> **Claim 1** If a is a root of a polynomial $p(x)$ with degree d, then $p(x) = (x-a)q(x)$ for a polynomial $q(x)$ with degree $d-1$.
>
> **Claim 2** A polynomial $p(x)$ of degree d with distinct roots $a_1, \ldots, a_d$ can be written as $p(x) = c(x-a_1)\cdots(x-a_d)$.

Claim 2 implies property 1. We must show that $a \neq a_i$ for $i = 1, \ldots d$ cannot be a root of $p(x)$. But this follows from claim 2, since $p(a) = c(a-a_1)\cdots(a-a_d) \neq 0$.

Proof of Claim 1

Dividing $p(x)$ by $(x-a)$ gives $p(x) = (x-a)q(x)+r(x)$, where $q(x)$ is the quotient and $r(x)$ is the remainder. The degree of $r(x)$ is necessarily smaller than the degree of the divisor $(x-a)$. Therefore $r(x)$ must have degree 0 and therefore is some constant c. But now substituting $x = a$, we get $p(a) = c$. But since a is a root, $p(a) = 0$. Thus $c = 0$ and therefore $p(x) = (x-a)q(x)$, thus showing that $(x-a)|p(x)$.

Claim 1 implies Claim 2

Proof by induction on d.

- Base Case: We must show that a polynomial $p(x)$ of degree 1 with root a_1 can be written as $p(x) = c(x-a_1)$. By Claim 1, we know that $p(x) = (x-a_1)q(x)$, where $q(x)$ has degree 0 and is therefore a constant.
- Inductive Hypothesis: A polynomial of degree $d-1$ with distinct roots $a_1, \ldots, a_{d-1}$ can be written as $p(x) = c(x-a_1)\cdots(x-a_{d-1})$.
- Inductive Step: Let $p(x)$ be a polynomial of of degree d with distinct roots $a_1, \cdots, a_d$. By Claim 1, $p(x) = (x-a_d)q(x)$ for some polynomial $q(x)$ of degree $d-1$. Since $0 = p(a_i) = (a_i-a_d)q(a_i)$ for all $i \neq d$ and $a_i - a_d \neq 0$ in this case, $q(a_i)$ must be equal to 0. Then $q(x)$ is a polynomial of degree $d-1$ with distinct roots $a_1, \ldots, a_{d-1}$. We can now apply the inductive assumption to $q(x)$ to write $q(x) = c(x-a_1)\cdots(x-a_{d-1})$. Substituting in $p(x) = (x-a_d)q(x)$, we finally obtain that $p(x) = c(x-a_1)\cdots(x-a_d)$.

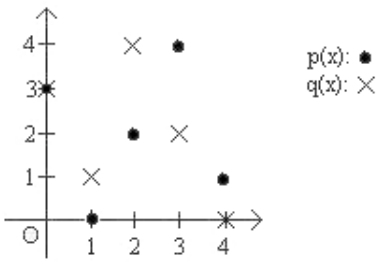
Finite Fields

Both property 1 and property 2 also hold when the values of the coefficients and the variable x are chosen from the complex numbers instead of the real numbers or even the rational numbers. They do not hold if the values are restricted to being natural numbers or integers. Let us try to understand this a little more closely. The only properties of numbers that we used in polynomial interpolation and in the proof of property 1 is that we can add, subtract, multiply and divide any pair of numbers as long as we are not dividing by 0. We cannot subtract two natural numbers and guarantee that the result is a natural number. And dividing two integers does not usually result in an integer.

But if we work with numbers modulo a prime m, then we can add, subtract, multiply and divide (by any non-zero number modulo m). To check this, recall that x has an inverse mod m if $\gcd(m,x) = 1$, so if m is prime *all* the numbers $\{1,\ldots,m-1\}$ have an inverse mod m. So both property 1 and property 2 hold if the coefficients and the variable x are restricted to take on values modulo m. This remarkable fact that these properties hold even when we restrict ourselves to a *finite* set of values is the key to several applications that we will presently see.

Let us consider an example of degree $d = 1$ polynomials modulo 5. Let $p(x) = 2x+3 \pmod 5$. The roots of this polynomial are all values x such that $2x+3 = 0 \pmod 5$ holds. Solving for x, we get that $2x = -3 = 2 \pmod 5$ or $x = 1 \pmod 5$. Note that this is consistent with property 1 since we got only 1 root of a degree 1 polynomial.

Now consider the polynomials $p(x) = 2x+3$ and $q(x) = 3x-2$ with all numbers reduced mod 5. We can plot the value of each polynomial y as a function of x in the x-y plane. Since we are working modulo 5, there are only 5 possible choices for x, and only 5 possible choices for y:



Notice that these two “lines" intersect in exactly one point, even though the picture looks nothing at all like lines in the Euclidean plane! Looking at these graphs it might seem remarkable that both property 1 and property 2 hold when we work modulo m for any prime number m. But as we stated above, all that was required for the proofs of property 1 and 2 was the ability to add, subtract, multiply and divide any pair of numbers (as long as we are not dividing by 0), and they hold whenever we work modulo a prime m.

When we work with numbers modulo a prime m, we say that we are working over a finite field, denoted by F_m or $GF(m)$ (for Galois Field). In order for a set to be called a field, it must satisfy certain axioms which are the building blocks that allow for these amazing properties and others to hold. If you would like to learn more about fields and the axioms they satisfy, you can visit Wikipedia's site and read the article on fields: `http://en.wikipedia.org/wiki/Field_\%28mathematics\%29`. While you are there, you can also read the article on Galois Fields and learn more about some of their applications and elegant properties which will not be covered in this lecture: `http://en.wikipedia.org/wiki/Galois_field`.

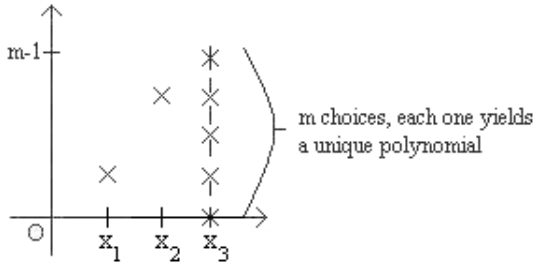
Counting

How many polynomials of degree (at most) 2 are there modulo m? This is easy: there are 3 coefficients, each of which can take on one of m values for a total of m^3. Writing $p(x) = a_d x^d + a_{d-1}x^{d-1} + \ldots + a_0$ by specifying its $d+1$ coefficients a_i is known as the coefficient representation of $p(x)$. Is there any other way to specify $p(x)$?

Sure, there is! Our polynomial of degree (at most) 2 is uniquely specified by its values at any three points, say $x = 0,1,2$. Once again each of these three values can take on one of m values, for a total of m^3 possibilities. In general, we can specify a degree d polynomial $p(x)$ by specifying its values at $d+1$ points, say $0,1,\ldots,d$. These $d+1$ values, $(y_0, y_1, \ldots, y_d)$ are called the value representation of $p(x)$. The coefficient representation

can be converted to the value representation by evaluating the polynomial at $0, 1, \ldots, d$. And as we've seen, polynomial interpolation can convert the value representation to the coefficient representation.

So if we are given three pairs $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ then there is a unique polynomial of degree 2 such that $p(x_i) = y_i$. But now, suppose we were only given two pairs $(x_1, y_1), (x_2, y_2)$; how many distinct degree 2 polynomials are there that go through these two points? Notice that there are exactly m choices for y_3, and for each choice there is a unique (and distinct) polynomial of degree 2 that goes through the three points $(x_1, y_1), (x_2, y_2), (x_3, y_3)$. It follows that there are exactly m polynomials of degree at most 2 that go through 2 points $(x_1, y_1), (x_2, y_2)$, as shown below:



What if you were only given one point (x_1, y_1)? Well, there are m^2 choices for y_2 and y_3, yielding m^2 polynomials of degree at most 2 that go through the point given. A pattern begins to emerge, as is summarized in the following table:

Polynomials of degree $\leq d$ over F_m	
# of points	# of polynomials
$d+1$	1
d	m
$d-1$	m^2
$\vdots$	$\vdots$
$d-k$	m^{k+1}
$\vdots$	$\vdots$
0	m^{d+1}

Note that the reason that we can count the number of polynomials in this setting is because we are working over a finite field. If we were working over an infinite field such as the rationals, there would be infinitely many polynomials of degree d that can go through d points! Think of a line, which has degree one. If you were just given one point, there would be infinitely many possibilities for the second point, each of which uniquely defines a line.

Secret Sharing

In the late 1950's and into the 1960's, during the Cold War, President Dwight D. Eisenhower approved instructions and authorized top commanding officers for the use of nuclear weapons under very urgent emergency conditions. Such measures were set up in order to defend the United States in case of an attack in which there was not enough time to confer with the President and decide on an appropriate response. This would allow for a rapid response in case of a Soviet attack on U.S. soil. This is a perfect situation in which a secret sharing scheme could be used to ensure that a certain number of officials must come together in order to successfully launch a nuclear strike, so that for example no single person has the power and control

over such a devastating and destructive weapon. Suppose the U.S. government finally decides that a nuclear strike can be initiated only if at least $k > 1$ major officials agree to it. We want to devise a scheme such that (1) any group of k of these officials can pool their information to figure out the launch code and initiate the strike but (2) no group of $k-1$ or fewer have any information about the launch code, even if they pool their knowledge. For example, they should not learn whether the secret is odd or even, a prime number, divisible by some number a, or the secret's least significant bit. How can we accomplish this?

Suppose that there are n officials indexed from 1 to n and the launch code is some natural number s. Let q be a prime number larger than n and s. We will work over $GF(q)$ from now on.

Now pick a random polynomial $P(x)$ of degree $k-1$ such that $P(0) = s$ and give $P(1)$ to the first official, $P(2)$ to the second,..., $P(n)$ to the n^{th}. Then

- Any k officials, having the values of the polynomial at k points, can use Lagrange interpolation to find P, and once they know what P is, they can compute $P(0) = s$ to learn the secret. Another way to say this is that any k officials have between them a value representation of the polynomial, which they can convert to the coefficient representation, which allows them to evaluate $P(0) = s$.
- Any group of $k-1$ officials has no information about s. So they know only $k-1$ points through which $P(x)$, an unknown polynomial of degree $k-1$ passes. They wish to reconstruct $P(0)$. But by our discussion in the previous section, for each possible value $P(0) = b$, there is a unique polynomial of degree $k-1$ that passes through the $k-1$ points of the $k-1$ officials as well as through $(0,b)$. Hence the secret could be any of the q possible values $\{0,1,\ldots,q-1\}$, so the officials have—in a very precise sense—no information about s. Another way of saying this is that the information of the officials is consistent with q different value representations, one for each possible value of the secret, and thus the officials have no information about s. (Note that this is the main reason we choose to work over finite fields rather than, say, over the real numbers, where the basic secret-sharing scheme would still work. Because there are only finitely many values in our field, we can quantify precisely how many remaining possibilities there are for the value of the secret, and show that this is the same as if the officials had no information at all.)

Example. Suppose you are in charge of setting up a secret sharing scheme, with secret $s = 1$, where you want to distribute $n = 5$ shares to 5 people such that any $k = 3$ or more people can figure out the secret, but two or fewer cannot. Let's say we are working over $GF(7)$ (note that $7 > s$ and $7 > n$) and you randomly choose the following polynomial of degree $k-1 = 2 : P(x) = 3x^2 + 5x + 1$ (here, $P(0) = 1 = s$, the secret). So you know everything there is to know about the secret and the polynomial, but what about the people that receive the shares? Well, the shares handed out are $P(1) = 2$ to the first official, $P(2) = 2$ to the second, $P(3) = 1$ to the third, $P(4) = 6$ to the fourth, and $P(5) = 3$ to the fifth official. Let's say that officials 3, 4, and 5 get together (we expect them to be able to recover the secret). Using Lagrange interpolation, they compute the following delta functions:

$$\Delta_3(x) = \frac{(x-4)(x-5)}{(3-4)(3-5)} = \frac{(x-4)(x-5)}{2} = 4(x-4)(x-5);$$
$$\Delta_4(x) = \frac{(x-3)(x-5)}{(4-3)(4-5)} = \frac{(x-3)(x-5)}{-1} = 6(x-3)(x-5);$$
$$\Delta_5(x) = \frac{(x-3)(x-4)}{(5-3)(5-4)} = \frac{(x-3)(x-4)}{2} = 4(x-3)(x-4).$$

They then compute the polynomial over $GF(7)$: $P(x) = (1)\Delta_3(x) + (6)\Delta_4(x) + (3)\Delta_5(x) = 3x^2 + 5x + 1$ (verify the computation!). Now they simply compute $P(0)$ and discover that the secret is 1.

Let's see what happens if two officials try to get together, say persons 1 and 5. They both know that the polynomial looks like $P(x) = a_2x^2 + a_1x + s$. They also know the following equations:

$$P(1) = a_2 + a_1 + s = 2$$
$$P(5) = 4a_2 + 5a_1 + s = 3$$

But that is all they have two equations with three unknowns, and thus they cannot find out the secret. This is the case no matter which two officials get together. Notice that since we are working over $GF(7)$, the two people could have guessed the secret ($0 \leq s \leq 6$) and constructed a unique degree 2 polynomial (by property 2). But the two people combined have the same chance of guessing what the secret is as they do individually. This is important, as it implies that two people have no more information about the secret than one person does.

Error Correcting Codes

We will consider two situations in which we wish to transmit information on an unreliable channel. The first is exemplified by the internet, where the information (say a file) is broken up into packets, and the unreliability is manifest in the fact that some of the packets are lost (or erased) during transmission. Moreover the packets are labeled so that the recipient knows exactly which packets were received and which were dropped. We will refer to such errors as erasure errors. See the figure below:

In the second situation, some of the packets are corrupted during transmission due to channel noise. Now the recipient has no idea which packets were corrupted and which were received unmodified:

In the above example, packets 1 and 4 are corrupted. These types of errors are called general errors. We will discuss methods of encoding messages, called error correcting codes, which are capable of correcting both erasure and general errors.

Assume that the information consists of n packets. We can assume without loss of generality that the contents of each packet is a number modulo q (denoted by $GF(q)$), where q is a prime. For example, the contents of the packet might be a 32-bit string and can therefore be regarded as a number between 0 and $2^{32}-1$; then we could choose q to be any prime larger than 2^{32}. The properties of polynomials over $GF(q)$ (i.e., with coefficients and values reduced modulo q) are the backbone of both error-correcting schemes. To see this, let us denote the message to be sent by $m_1,\ldots,m_n$ and make the following crucial observations:

1) There is a unique polynomial $P(x)$ of degree $n-1$ such that $P(i)=m_i$ for $1\leq i\leq n$ (i.e., $P(x)$ contains all of the information about the message, and evaluating $P(i)$ gives the contents of the *i-th* packet).

2) The message to be sent is now $m_1=P(1),\ldots,m_n=P(n)$. We can generate additional packets by evaluating $P(x)$ at additional points $n+1,n+2,\ldots,n+j$ (remember, our transmitted message must be redundant, i.e., it must contain more packets than the original message to account for the lost or corrupted packets). Thus the transmitted message is $c_1=P(1),c_2=P(2),\ldots,c_{n+j}=P(n+j)$. Since we are working modulo q, we must make sure that $n+j\leq q$, but this condition does not impose a serious constraint since q is very large.

Erasure Errors

Here we consider the setting of packets being transmitted over the internet. In this setting, the packets are labeled and so the recipient knows exactly which packets were dropped during transmission. One additional observation will be useful:

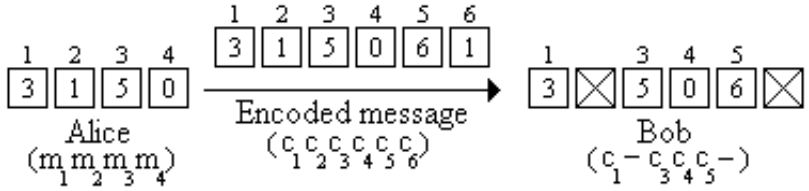
3) By Property 2 in Note 8, we can uniquely reconstruct $P(x)$ from its values at any n distinct points, since it has degree $n-1$. This means that $P(x)$ can be reconstructed from any n of the transmitted packets. Evaluating this reconstructed polynomial $P(x)$ at $x = 1, \ldots, n$ yields the original message $m_1, \ldots, m_n$.

Recall that in our scheme, the transmitted message is $c_1 = P(1), c_2 = P(2), \ldots, c_{n+j} = P(n+j)$. Thus, if we hope to be able to correct k errors, we simply need to set $j = k$. The encoded message will then consist of $n+k$ packets.

Example

Suppose Alice wants to send Bob a message of $n = 4$ packets and she wants to guard against $k = 2$ lost packets. Then, assuming the packets can be coded up as integers between 0 and 6, Alice can work over $GF(7)$ (since $7 \geq n+k = 6$). Suppose the message that Alice wants to send to Bob is $m_1 = 3$, $m_2 = 1$, $m_3 = 5$, and $m_4 = 0$. She interpolates to find the unique polynomial of degree $n-1 = 3$ described by these 4 points: $P(x) = x^3 + 4x^2 + 5$ (verify that $P(i) = m_i$ for $1 \leq i \leq 4$).

Since $k = 2$, Alice must evaluate $P(x)$ at 2 extra points: $P(5) = 6$ and $P(6) = 1$. Now, Alice can transmit the encoded message which consists of $n+k = 6$ packets, where $c_j = P(j)$ for $1 \leq j \leq 6$. So $c_1 = P(1) = 3$, $c_2 = P(2) = 1$, $c_3 = P(3) = 5$, $c_4 = P(4) = 0$, $c_5 = P(5) = 6$, and $c_6 = P(6) = 1$. Suppose packets 2 and 6 are dropped, in which case we have the following situation:



From the values that Bob received (3, 5, 0, and 6), he uses Lagrange interpolation and computes the following delta functions:

$$\Delta_1(x) = \frac{(x-3)(x-4)(x-5)}{-24}$$
$$\Delta_3(x) = \frac{(x-1)(x-4)(x-5)}{4}$$
$$\Delta_4(x) = \frac{(x-1)(x-3)(x-5)}{-3}$$
$$\Delta_5(x) = \frac{(x-1)(x-3)(x-4)}{8}.$$

He then reconstructs the polynomial $P(x) = (3)\Delta_1(x) + (5)\Delta_3(x) + (0)\Delta_4(x) + (6)\Delta_5(x) = x^3 + 4x^2 + 5$. Bob then evaluates $m_2 = P(2) = 1$, which is the packet that was lost from the original message. More generally, no matter which two packets were dropped, following the same method Bob could still have reconstructed $P(x)$ and thus the original message.

Let us consider what would happen if Alice sent one fewer packet. If Alice only sent c_j for $1 \leq j \leq n+k-1$, then with k erasures, Bob would only receive c_j for $n-1$ distinct values j. Thus, Bob would not be able to reconstruct $P(x)$ (since there are exactly q polynomials of degree at most $n-1$ that agree with the $n-1$ packets which Bob received). This error-correcting scheme is therefore optimal: it can recover the n characters of the transmitted message from any n received characters, but recovery from any fewer characters is impossible.

Polynomial Interpolation

Let us take a brief digression to discuss another method of polynomial interpolation which will be useful in handling general errors. The goal of the algorithm will be to take as input $d+1$ pairs $(x_1,y_1),\cdots,(x_{d+1},y_{d+1})$, and output the polynomial $p(x)=a_dx^d+\cdots+a_1x+a_0$ such that $p(x_i)=y_i$ for $i=1$ to $d+1$.

The first step of the algorithm is to write a system of $d+1$ linear equations in $d+1$ variables: the coefficients of the polynomial $a_0,\ldots,a_d$. Each equation is obtained by fixing x to be one of $d+1$ values: $x_1,\cdots,x_{d+1}$. Note that in $p(x)$, x is a variable and $a_0,\ldots,a_d$ are fixed constants. In the equations below, these roles are swapped: x_i is a fixed constant and $a_0,\ldots,a_d$ are variables. For example, the *i-th* equation is the result of fixing x to be x_i: $a_dx_i^d+a_{d-1}x_i^{d-1}+\ldots+a_0=y_i$.

Now solving these equations gives the coefficients of the polynomial $p(x)$. For example, given the 3 pairs $(-1,2)$, $(0,1)$, and $(2,5)$, we will construct the degree 2 polynomial $p(x)$ which goes through these points. The first equation says $a_2(-1)^2+a_1(-1)+a_0=2$. Simplifying, we get $a_2-a_1+a_0=2$. Similarly, the second equation says $a_2(0)^2+a_1(0)+a_0=1$, or $a_0=1$. And the third equation says $a_2(2)^2+a_1(2)+a_0=5$ So we get the following system of equations:

$$\begin{aligned} a_2-a_1+a_0&=2\\ a_0&=1\\ 4a_2+2a_1+a_0&=5 \end{aligned}$$

Substituting for a_0 and multiplying the first equation by 2 we get:

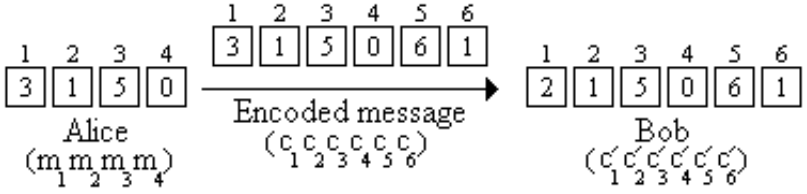
$$\begin{aligned} 2a_2-2a_1&=2\\ 4a_2+2a_1&=4 \end{aligned}$$

Then, adding the two equations we find that $6a_2=6$, so $a_2=1$, and plugging back in we find that $a_1=0$. Thus, we have determined the polynomial $p(x)=x^2+1$. To justify this method more carefully, we must show that the equations always have a solution and that it is unique. This involves showing that a certain determinant is non-zero, which we will leave as an exercise.

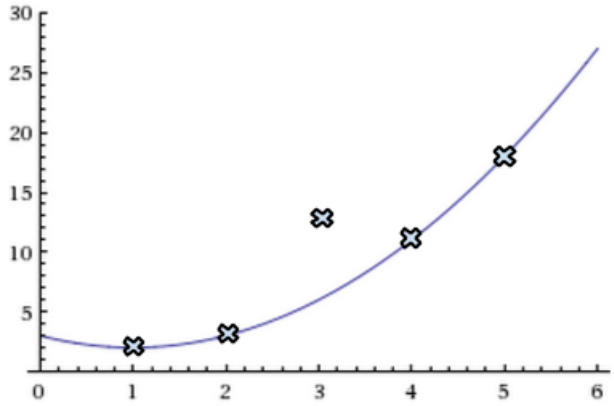
General Errors

Now let us return to general errors. General errors are much more challenging to correct than erasure errors. This is because packets are corrupted, not erased and Bob no longer knows which packets are correct. As we shall see shortly, Alice can still guard against k general errors, at the expense of transmitting only $2k$ additional packets or characters (only twice as many as in the erasure case). Thus the encoded message is $c_1,\ldots,c_{n+2k}$ where $c_j=P(j)$ for $1\le j\le n+2k$. This means that at least $n+k$ of these characters are received uncorrupted by Bob.

For example, if Alice wishes to send $n=4$ characters to Bob via a modem in which $k=1$ of the characters is corrupted, she must redundantly send an encoded message consisting of 6 characters. Suppose she wants to transmit the same message as above, and that c_1 is corrupted and changed to $r_1=2$. This scenario can be visualized in the following figure:



Bob's goal is to reconstruct $P(x)$ from the $n+2k$ received characters $r_1, \cdots, r_{n+2k}$. He knows that $P(i)$ must equal r_i on at least $n+k$ points (since only k points are corrupted), but he does not know which of the $n+k$ values are correct. As an example, consider a possible scenario depicted in the picture below- the points represent the message received from Alice, and the line represents $P(x)$. In this example, $n=3$, $k=1$, and the third packet is corrupted. Bob does not know the index at which the message and the polynomial deviate:



Bob attempts to construct $P(x)$ by searching for a polynomial $P'(x)$ with the following property: $P'(i) = r_i$ for at least $n+k$ distinct values of i between 1 and $n+2k$. Of course, $P(x)$ is one such polynomial. It turns out that $P(x)$ is actually the only polynomial with the desired property. Therefore, $P'(x)$ must equal $P(x)$.

Finding $P(x)$ efficiently requires a remarkable idea, which is just about simple enough to be described here. Suppose packets $e_1, \ldots, e_k$ are corrupted. Define the degree k polynomial $E(x)$ to be $(x-e_1)\cdots(x-e_k)$. Let us make a simple but crucial observation: $P(i)E(i) = r_iE(i)$ for $1 \leq i \leq n+2k$ (this is true at points i at which no error occurred since $P(i) = r_i$, and trivially true at points i at which an error occurred since $E(i) = 0$).

This observation forms the basis of a very clever algorithm invented by Berlekamp and Welch. Looking more closely at these equalities, we will show that they are $n+2k$ linear equations in $n+2k$ unknowns. The unknowns correspond to the coefficients of $E(x)$ and $Q(x)$ (where we define $Q(x) = P(x)E(x)$). Once $Q(x)$ and $E(x)$ are known, we can divide $Q(x)$ by $E(x)$ to obtain $P(x)$.

Since $Q(x)$ is a polynomial of degree $n+k-1$, it can be described by $n+k$ coefficients. $E(x)$ is a degree k polynomial, but its definition implies that its first coefficient must be 1. It can therefore be described by k coefficients:

$$Q(x) = a_{n+k-1}x^{n+k-1} + \cdots + a_1x + a_0$$
$$E(x) = x^k + b_{k-1}x^{k-1} + \cdots + b_1x + b_0$$

As seen in the interpolation method above, once we fix a value i for x, $Q(i)$ and $E(i)$ are linear functions of $a_{n+k-1}, \cdots, a_0$ and $b_{k-1}, \cdots, b_0$ respectively. The received value r_i is also fixed. Therefore the equation $Q(i) = r_iE(i)$ is a linear equation in the $n+2k$ unknowns $a_{n+k-1}, \ldots, a_0$ and $b_{k-1}, \ldots, b_0$. We thus have $n+2k$ linear equations, one for each value of i, and $n+2k$ unknowns. We can solve these equations and get $E(x)$ and $Q(x)$. We can then compute the ratio $\frac{Q(x)}{E(x)}$ to obtain $P(x)$.

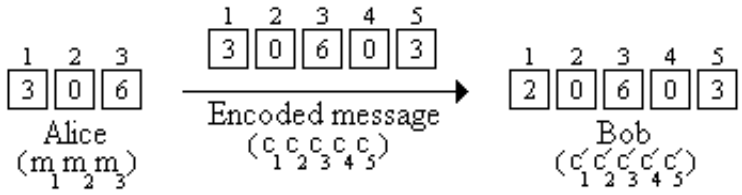
Example

Suppose we are working over $GF(7)$ and Alice wants to send Bob the $n = 3$ characters "3," "0," and "6" over a modem. Turning to the analogy of the English alphabet, this is equivalent to using only the first 7 letters of the alphabet, where $a = 0, b = 1, \ldots, g = 6$. So the message which Alice wishes for Bob to receive is "dag". Then Alice interpolates to find the polynomial

$$P(x) = x^2 + x + 1,$$

which is the unique polynomial of degree 2 such that $P(1) = 3$, $P(2) = 0$, and $P(3) = 6$.

Suppose that $k = 1$ character is corrupted, so she needs to transmit the $n + 2k = 5$ characters $P(1) = 3$, $P(2) = 0$, $P(3) = 6$, $P(4) = 0$, and $P(5) = 3$ to Bob. Suppose $P(1)$ is corrupted, so he receives 2 instead of 3 (i.e., Alice sends the encoded message "dagad" but Bob instead receives "cagad"). Summarizing, we have the following situation:



Let $E(x) = x + b_0$ be the error-locator polynomial—remember, Bob doesn't know what b_0 is yet since he doesn't know where the error occurred. Let $Q(x) = a_3x^3 + a_2x^2 + a_1x + a_0$. Now Bob just substitutes $x = 1, x = 2, \cdots, x = 5$ into $Q(x) = r_xE(x)$ and simplifies to get five linear equations in five unknowns. Recall that we are working modulo 7 and that $r_i = c_i'$ is the value Bob received for the *i-th* character.

The first equation will be $a_3 + a_2 + a_1 + a_0 = 2(1 + b_0)$, which simplifies to $a_3 + a_2 + a_1 + a_0 + 5b_0 = 2$. Bob can determine the remaining equations in the same manner, obtaining:

$$\begin{aligned}
a_3 + a_2 + a_1 + a_0 + 5b_0 &= 2 \\
a_3 + 4a_2 + 2a_1 + a_0 &= 0 \\
6a_3 + 2a_2 + 3a_1 + a_0 + b_0 &= 4 \\
a_3 + 2a_2 + 4a_1 + a_0 &= 0 \\
6a_3 + 4a_2 + 5a_1 + a_0 + 4b_0 &= 1
\end{aligned}$$

Bob then solves this linear system and finds that $a_3 = 1$, $a_2 = 0$, $a_1 = 0$, $a_0 = 6$, and $b_0 = 6$ (all mod 7). (As a check, this implies that $E(x) = x + 6 = x - 1$, so the location of the error is position $e_1 = 1$, which is correct since the first character was corrupted from a "d" to a "c".) This gives him the polynomials $Q(x) = x^3 + 6$ and $E(x) = x - 1$. He can then find $P(x)$ by computing the quotient $P(x) = \frac{Q(x)}{E(x)} = \frac{x^3+6}{x-1} = x^2 + x + 1$. Bob notices that the first character was corrupted (since $e_1 = 1$), so now that he has $P(x)$, he just computes $P(1) = 3 =$ "d" and obtains the original, uncorrupted message "dag".

Finer points

Two points need further discussion. How do we know that the $n + 2k$ equations are consistent? What if they have no solution? This is simple. The equations must be consistent since $Q(x) = P(x)E(x)$ together with the error locator polynomial $E(x)$ gives a solution.

A more interesting question is this: how do we know that the $n+2k$ equations are independent, i.e., how do we know that there aren't other spurious solutions in addition to the real solution that we are looking for? Put more mathematically, how do we know that the solution $Q'(x)$ and $E'(x)$ that we reconstruct satisfies the property that $E'(x)$ divides $Q'(x)$ and that $\frac{Q'(x)}{E'(x)} = \frac{Q(x)}{E(x)} = P(x)$?

We claim that $Q(x)E'(x) = Q'(x)E(x)$ for $1 \leq x \leq n+2k$. Since the degree of both $Q(x)E'(x)$ and $Q'(x)E(x)$ is $n+2k-1$ and they are equal at $n+2k$ points, it follows from Property 2 of Note 7 that they are the same polynomial. Rearranging, we get $\frac{Q'(x)}{E'(x)} = \frac{Q(x)}{E(x)} = P(x)$.

Why is the claim above true? Based on our method of obtaining $Q'(x)$ and $E'(x)$, we know that $Q'(i) = r_i E'(i)$ and $Q(i) = r_i E(i)$. Now assume $E(i)$ is 0. Then $Q(i)$ is also 0, so both $Q(i)E'(i)$ and $Q'(i)E(i)$ are 0 and the claim holds. The same reasoning applies when $E'(i)$ is 0. If both $E(i)$ and $E'(i)$ are not 0, we can rearrange the above equality to obtain $\frac{Q'(i)}{E'(i)} = \frac{Q(i)}{E(i)}$, which implies the claim.

Infinity and Countability

Cardinality

How can we determine whether two sets have the same *cardinality* (or "size")? The answer to this question, reassuringly, lies in early grade school memories: by demonstrating a *pairing* between elements of the two sets. More formally, we need to demonstrate a *bijection* f between the two sets. The bijection sets up a one-to-one correspondence, or pairing, between elements of the two sets. We know how this works for finite sets. In this lecture, we will see what it tells us about *infinite* sets.

Are there more natural numbers $\mathbb{N}$ than there are positive integers $\mathbb{Z}^+$? It is tempting to answer yes, since every positive integer is also a natural number, but the natural numbers have one extra element $0 \notin \mathbb{Z}^+$. Upon more careful observation, though, we see that we can generate a mapping between the natural numbers and the positive integers as follows:

$\mathbb{N}$	0	1	2	3	4	5	...
$\downarrow$		$\searrow$	$\searrow$	$\searrow$	$\searrow$	$\searrow$	$\searrow$
$\mathbb{Z}^+$		1	2	3	4	5	6 ...

Why is this mapping a bijection? Clearly, the function $f : \mathbb{N} \to \mathbb{Z}^+$ is onto because every positive integer is hit. And it is also one-to-one because no two natural numbers have the same image. (The image of n is $f(n) = n+1$, so if $f(n) = f(m)$ then we must have $n = m$.) Since we have shown a bijection between $\mathbb{N}$ and $\mathbb{Z}^+$, this tells us that there are as many natural numbers as there are positive integers! Informally, we have proved that "$\infty + 1 = \infty$."

What about the set of *even* natural numbers $2\mathbb{N} = \{0, 2, 4, 6, ...\}$? In the previous example the difference was just one element. But in this example, there seem to be twice as many natural numbers as there are even natural numbers. Surely, the cardinality of $\mathbb{N}$ must be larger than that of $2\mathbb{N}$ since $\mathbb{N}$ contains all of the odd natural numbers as well. Though it might seem to be a more difficult task, let us attempt to find a bijection between the two sets using the following mapping:

$\mathbb{N}$	0	1	2	3	4	5	...
$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	
$2\mathbb{N}$	0	2	4	6	8	10	...

The mapping in this example is also a bijection. f is clearly one-to-one, since distinct natural numbers get mapped to distinct even natural numbers (because $f(n) = 2n$). f is also onto, since every n in the range is hit: its pre-image is $\frac{n}{2}$. Since we have found a bijection between these two sets, this tells us that in fact $\mathbb{N}$ and $2\mathbb{N}$ have the same cardinality!

What about the set of all integers, $\mathbb{Z}$? At first glance, it may seem obvious that the set of integers is larger

than the set of natural numbers, since it includes negative numbers. However, as it turns out, it is possible to find a bijection between the two sets, meaning that the two sets have the same size! Consider the following mapping:

$0 \to 0, \quad 1 \to -1, \quad 2 \to 1, \quad 3 \to -2, \quad 4 \to 2, \quad \ldots, \quad 124 \to 62, \quad \ldots$

In other words, our function is defined as follows:

$$f(x) = \begin{cases} \frac{x}{2} & \text{if } x \text{ is even} \\ \frac{-(x+1)}{2} & \text{if } x \text{ is odd} \end{cases}$$

We will prove that this function $f : \mathbb{N} \to \mathbb{Z}$ is a bijection, by first showing that it is one-to-one and then showing that it is onto.

Proof (one-to-one): Suppose $f(x) = f(y)$. Then they both must have the same sign. Therefore either $f(x) = \frac{x}{2}$ and $f(y) = \frac{y}{2}$, or $f(x) = \frac{-(x+1)}{2}$ and $f(y) = \frac{-(y+1)}{2}$. In the first case, $f(x) = f(y) \Rightarrow \frac{x}{2} = \frac{y}{2} \Rightarrow x = y$. Hence $x = y$. In the second case, $f(x) = f(y) \Rightarrow \frac{-(x+1)}{2} = \frac{-(y+1)}{2} \Rightarrow x = y$. So in both cases $f(x) = f(y) \Rightarrow x = y$, so f is injective.

Proof (onto): If $y \in \mathbb{Z}$ is non-negative, then $f(2y) = y$. Therefore, y has a pre-image. If y is negative, then $f(-(2y+1)) = y$. Therefore, y has a pre-image. Thus every $y \in \mathbb{Z}$ has a preimage, so f is onto.

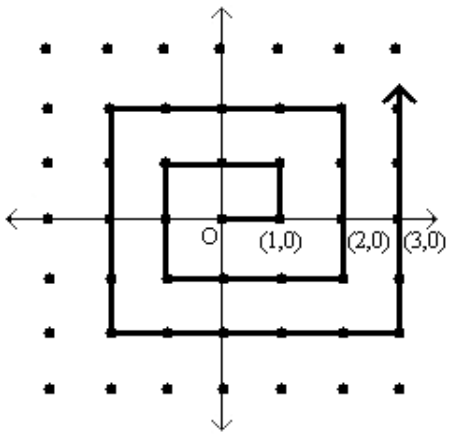
Since f is a bijection, this tells us that $\mathbb{N}$ and $\mathbb{Z}$ have the same size.

Now for an important definition. We say that a set S is **countable** if there is a bijection between S and $\mathbb{N}$ or some subset of $\mathbb{N}$. Thus any finite set S is countable (since there is a bijection between S and the subset $\{0, 1, 2, \ldots, m-1\}$, where $m = |S|$ is the size of S). And we have already seen three examples of countable infinite sets: $\mathbb{Z}^+$ and $2\mathbb{N}$ are obviously countable since they are themselves subsets of $\mathbb{N}$; and $\mathbb{Z}$ is countable because we have just seen a bijection between it and $\mathbb{N}$.

What about the set of all rational numbers? Recall that $\mathbb{Q} = \{\frac{x}{y} \mid x, y \in \mathbb{Z},\ y \neq 0\}$. Surely there are more rational numbers than natural numbers? After all, there are infinitely many rational numbers between any two natural numbers. Surprisingly, the two sets have the same cardinality! To see this, let us introduce a slightly different way of comparing the cardinality of two sets.

If there is a one-to-one function $f : A \to B$, then the cardinality of A is less than or equal to that of B. Now to show that the cardinality of A and B are the same we can show that $|A| \leq |B|$ and $|B| \leq |A|$. This corresponds to showing that there is a one-to-one function $f : A \to B$ and a one-to-one function $g : B \to A$. The existence of these two one-to-one functions implies that there is a bijection $h : A \to B$, thus showing that A and B have the same cardinality. The proof of this fact, which is called the Cantor-Bernstein theorem, is actually quite hard, and we will skip it here.

Back to comparing the natural numbers and the rationals. First it is obvious that $|\mathbb{N}| \leq |\mathbb{Q}|$ because $\mathbb{N} \subseteq \mathbb{Q}$. So our goal now is to prove that also $|\mathbb{Q}| \leq |\mathbb{N}|$. To do this, we must exhibit an injection $f : \mathbb{Q} \to \mathbb{N}$. The following picture of a spiral conveys the idea of this injection:



Each rational number $\frac{a}{b}$ (written in its lowest terms, so that $\gcd(a,b) = 1$) is represented by the point (a,b) in the infinite two-dimensional grid shown (which corresponds to $\mathbb{Z} \times \mathbb{Z}$, the set of all pairs of integers). Note that not all points on the grid are valid representations of rationals: e.g., all points on the x-axis have $b = 0$ so none are valid (except for $(0,0)$, which we take to represent the rational number 0); and points such as $(2,8)$ and $(-1,-4)$ are not valid either as the rational number $\frac{1}{4}$ is represented by $(1,4)$. But $\mathbb{Z} \times \mathbb{Z}$ certainly contains all rationals under this representation, so if we come up with an injection from $\mathbb{Z} \times \mathbb{Z}$ to $\mathbb{N}$ then this will also be an injection from $\mathbb{Q}$ to $\mathbb{N}$ (why?).

The idea is to map each pair (a,b) to its position along the spiral, starting at the origin. (Thus, e.g.,$(0,0) \to 0$, $(1,0) \to 1$, $(1,1) \to 2$, $(0,1) \to 3$, and so on.) This mapping certainly maps every rational number to a natural number, because every rational appears somewhere (exactly once) in the grid, and the spiral hits every point in the grid. Why is this mapping an injection? Well, we just have to check that no two rational numbers map to the same natural number. But that is true because no two pairs lie at the same position on the spiral. (Note that the mapping is *not* onto because some positions along the spiral do not correspond to valid representations of rationals; but that is fine.)

This tells us that $|\mathbb{Q}| \leq |\mathbb{N}|$. Since also $|\mathbb{N}| \leq |\mathbb{Q}|$, as we observed earlier, by the Cantor-Bernstein Theorem $\mathbb{N}$ and $\mathbb{Q}$ have the same cardinality.

Our next example concerns the set of all binary strings (of any finite length), denoted $\{0,1\}^*$. Despite the fact that this set contains strings of unbounded length, it turns out to have the same cardinality as $\mathbb{N}$. To see this, we set up a direct bijection $f : \{0,1\}^* \to \mathbb{N}$ as follows. Note that it suffices to *enumerate* the elements of $\{0,1\}^*$ in such a way that each string appears exactly once in the list. We then get our bijection by setting $f(n)$ to be the nth string in the list. How do we enumerate the strings in $\{0,1\}^*$? Well, it's natural to list them in increasing order of length, and then (say) in *lexicographic* order (or, equivalently, numerically increasing order when viewed as binary numbers) within the strings of each length. This means that the list would look like

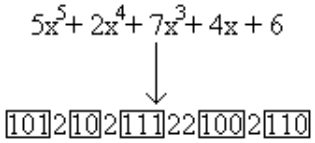
$$\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, 1000, \ldots,$$

where ε denotes the empty string (the only string of length 0). It should be clear that this list contains each binary string once and only once, so we get a bijection with $\mathbb{N}$ as desired.

Our final countable example is the set of all polynomials with natural number coefficients, which we denote $\mathbb{N}(x)$. To see that this set is countable, we will make use of (a variant of) the previous example. Note first that, by essentially the same argument as for $\{0,1\}^*$, we can see that the set of all *ternary* strings $\{0,1,2\}^*$ (that is, strings over the alphabet $\{0,1,2\}$) is countable. To see that $\mathbb{N}(x)$ is countable, it therefore suffices to exhibit an injection $f : \mathbb{N}(x) \to \{0,1,2\}^*$, which in turn will give an injection from $\mathbb{N}(x)$ to $\mathbb{N}$. (It is obvious that there exists an injection from $\mathbb{N}$ to $\mathbb{N}(x)$, since each natural number n is itself trivially a polynomial,

namely the constant polynomial n itself.)

How do we define f? Let's first consider an example, namely the polynomial $p(x) = 5x^5 + 2x^4 + 7x^3 + 4x + 6$. We can list the coefficients of $p(x)$ as follows: $(5,2,7,0,4,6)$. We can then write these coefficients as binary strings: $(101_2, 10_2, 111_2, 0_2, 100_2, 110_2)$. Now, we can construct a ternary string where a "2" is inserted as a separator between each binary coefficient (ignoring coefficients that are 0). Thus we map $p(x)$ to a ternary string as illustrated below:

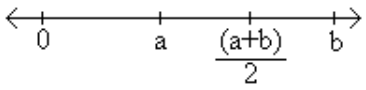


It is easy to check that this is an injection, since the original polynomial can be uniquely recovered from this ternary string by simply reading off the coefficients between each successive pair of 2's. (Notice that this mapping $f : \mathbb{N}(x) \to \{0,1,2\}^*$ is not onto (and hence not a bijection) since many ternary strings will not be the image of any polynomials; this will be the case, for example, for any ternary strings that contain binary subsequences with leading zeros.)

Hence we have an injection from $\mathbb{N}(x)$ to $\mathbb{N}$, so $\mathbb{N}(x)$ is countable.

Cantor's Diagonalization

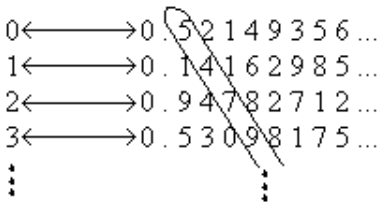
So we have established that $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ all have the same cardinality. What about the real numbers, the set of all points on the real line? Surely they are countable too? After all, the rational numbers, like the real numbers, are dense (i.e., between any two rational numbers there is a rational number):



In fact, between any two *real* numbers there is always a rational number. It is really surprising, then, that there are more real numbers than rationals. That is, there is no bijection between the rationals (or the natural numbers) and the reals. In fact, we will show something even stronger: even the real numbers in the interval $[0,1]$ are uncountable!

Recall that a real number can be written out in an infinite decimal expansion. A real number in the interval $[0,1]$ can be written as $0.d_1d_2d_3...$ Note that this representation is not unique; for example, $1 = 0.999...$[1]; for definiteness we shall assume that every real number is represented as a recurring decimal where possible (i.e., we choose the representation $.999\ldots$ rather than 1).

Cantor's Diagonalization Proof: Suppose towards a contradiction that there is a bijection $f : \mathbb{N} \to \mathbb{R}[0,1]$. Then, we can enumerate the infinite list as follows:



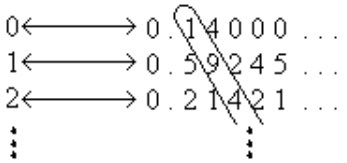
The number circled in the diagonal is some real number $D = 0.5479\ldots$, since it is an infinite decimal

[1] To see this, write $x = .999\ldots$. Then $10x = 9.999\ldots$, so $9x = 9$, and thus $x = 1$.

expansion. Now consider the real number s obtained by modifying every digit of D, say by replacing each digit d with $d+2$ mod 10; thus in our example above, $s = 0.7691\ldots$. We claim that s does not occur in our infinite list of real numbers. Suppose for contradiction that it did as the n^{th} number in the list. But the nth number's n^{th} digit is the same as the n^{th} digit of D and thus is different from the n^{th} digit of s: the n^{th} digit of s is the n^{th} digit of r plus 2 mod 10. So we have a real number s that is not in the range of f. But this contradicts the assertion that f is a bijection. Thus the real numbers are not countable.

Let us remark that the reason that we modified each digit by adding 2 mod 10 as opposed to adding 1 is that the same real number can have two decimal expansions; for example $0.999... = 1.000....$ But if two real numbers differ by more than 1 in any digit they cannot be equal. Thus we are completely safe in our assertion.

With Cantor's diagonalization method, we proved that $\mathbb{R}$ is uncountable. What happens if we apply the same method to $\mathbb{Q}$, in a (futile) attempt to show the rationals are uncountable? Well, suppose for a contradiction that our bijective function $f : \mathbb{N} \to \mathbb{Q}[0,1]$ produces the following mapping:

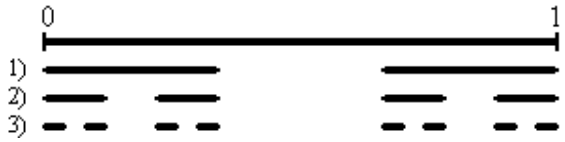


This time, let us consider the number q obtained by modifying every digit of the diagonal, say by replacing each digit d with $d+2$ mod 10. Then in the above example q = 0.316..., and we want to try to show that it does not occur in our infinite list of rational numbers. However, we do not know if q is rational (in fact, it is extremely unlikely for the decimal expansion of q to be periodic). This is why the method fails when applied to the rationals. When dealing with the reals, the modified diagonal number was guaranteed to be a real number.

The Cantor Set

Please read on only if interested.

The Cantor set is a remarkable set construction involving the real numbers in the interval $[0,1]$. The set is defined by repeatedly removing the middle thirds of line segments infinitely many times, starting with the original interval. For example, the first iteration would involve the removal of the interval $(\frac{1}{3}, \frac{2}{3})$, leaving $[0, \frac{1}{3}] \cup [\frac{2}{3}, 1]$. The first three iterations are illustrated below:



The Cantor set contains all points that have *not* been removed: $C = \{x : x \text{ not thrown out}\}$. How much of the original unit interval is left after this process is repeated infinitely? Well, we start with 1, and after the first iteration we remove $\frac{1}{3}$ of the interval, leaving us with $\frac{2}{3}$. For the second iteration, we keep $\frac{2}{3} \times \frac{2}{3}$ of the original interval. As we repeat the iterations infinitely, we are left with:

$$1 \longrightarrow \tfrac{2}{3} \longrightarrow \tfrac{2}{3} \times \tfrac{2}{3} \longrightarrow \tfrac{2}{3} \times \tfrac{2}{3} \times \tfrac{2}{3} \longrightarrow \cdots \longrightarrow \lim_{n \to \infty} (\tfrac{2}{3})^n = 0$$

According to the calculations, we have removed everything from the original interval! Does this mean that the Cantor set is empty? No, it doesn't. What it means is that the *measure* of the Cantor set is zero; the Cantor set consists of isolated points and does not contain any non-trivial intervals. In fact, not only is the

Cantor set non-empty, it is uncountable![2]

To see why, let us first make a few observations about ternary strings. In ternary notation, all strings consist of digits (called "trits") from the set $\{0,1,2\}$. All real numbers in the interval $[0,1]$ can be written in ternary notation. (E.g., $\frac{1}{3}$ can be written as 0.1, or equivalently as $0.0222\ldots$, and $\frac{2}{3}$ can be written as 0.2 or as $0.1222\ldots$.) Thus, in the first iteration, the middle third removed contains all ternary numbers of the form 0.1xxxxx. The ternary numbers left after the first removal can all be expressed either in the form 0.0xxxxx... or 0.2xxxxx... (We have to be a little careful here with the endpoints of the intervals; but we can handle them by writing $\frac{1}{3}$ as $0.02222\ldots$ and $\frac{2}{3}$ as 0.2.) The second iteration removes ternary numbers of the form 0.01xxxxx and 0.21xxxxx (i.e., any number with 1 in the second position). The third iteration removes 1's in the third position, and so on. Therefore, what remains is all ternary numbers with only 0's and 2's. Thus we have shown that

$$C = \{x \in [0,1] : x \text{ has a ternary representation consisting only of 0's and 2's}\}.$$

Finally, using this characterization, we can set up an *onto* map f from C to $[0,1]$. Since we already know that $[0,1]$ is uncountable, this implies that C is uncountable also. The map f is defined as follows: for $x \in C$, $f(x)$ is defined as the binary decimal obtained by dividing each digit of the ternary representation of x by 2. Thus, for example, if $x = 0.0220$ (in ternary), then $f(x)$ is the binary decimal 0.0110). But the set of all binary decimals $0.xxxxx\ldots$ is in 1-1 correspondence with the real interval $[0,1]$, and the map f is onto because every binary decimal is the image of some ternary string under f (obtained by doubling every binary digit).[3] This completes the proof that C is uncountable.

Power Sets and Higher Orders of Infinity (Optional)

Please read on only if interested.
Let S be any set. Then the *power set* of S, denoted by $\mathscr{P}(S)$, is the set of all subsets of S. More formally, it is defined as: $\mathscr{P}(S) = \{T : T \subseteq S\}$. For example, if $S = \{1,2,3\}$, then $\mathscr{P}(S) = \{\{\}, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$.

What is the cardinality of $\mathscr{P}(S)$? If $|S| = k$ is finite, then $|\mathscr{P}(S)| = 2^k$. To see this, let us think of each subset of S corresponding to a k bit string. In the example above the subset $\{1,3\}$ corresponds to the string 101. A 1 in the i^{th} position indicates that the i^{th} element of S is in the subset and a 0 indicates that it is not. Now the number of binary strings of length k is 2^k, since there are two choices for each bit position. Thus $|\mathscr{P}(S)| = 2^k$. So for finite sets S, the cardinality of the power set of S is exponentially larger than the cardinality of S. What about infinite (countable) sets? We claim that there is no bijection from S to $\mathscr{P}(S)$, so $\mathscr{P}(S)$ is not countable. Thus for example the set of all subsets of natural numbers is not countable, even though the set of natural numbers itself is countable.

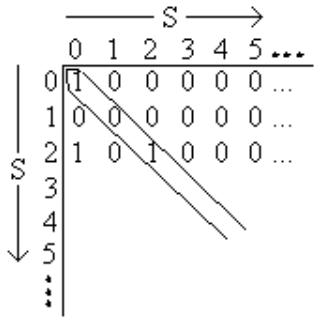
Theorem: Let S be countably infinite. Then $|\mathscr{P}(S)| > |S|$.

Proof: Suppose towards a contradiction that there is a bijection $f : S \to \mathscr{P}(S)$. Recall that we can represent a subset by a binary string, with one bit for each element of S. (So, since S is infinite, the string will be infinitely long. Contrast the case of $\{0,1\}^*$ discussed earlier, which consists of all binary strings of *finite*

[2] It's actually easy to see that C contains at least countably many points, namely the endpoints of the intervals in the construction—i.e., numbers such as $\frac{1}{3}$, $\frac{2}{3}$, $\frac{1}{9}$, $\frac{1}{27}$ etc. It's less obvious that C also contains various other points, such as $\frac{1}{4}$ and $\frac{3}{10}$. (Why?)

[3] Note that f is *not* injective; for example, the ternary strings $0.20222\ldots$ and 0.22 map to binary strings $0.10111\ldots$ and 0.11 respectively, which denote the same real number. Thus f is not a bijection. However, the current proof shows that the cardinality of C is at least that of $[0,1]$, while it is obvious that the cardinality of C is at most that of $[0,1]$ since $C \subset [0,1]$. Hence C has the same cardinality as $[0,1]$ (and as $\mathbb{R}$).

length.) Consider the following diagonalization picture in which the function f maps natural numbers x to binary strings which correspond to subsets of S (e.g., $2 \to 10100... = \{0,2\}$):



In this case, we have assigned the following mapping: $0 \to \{0\}$, $1 \to \{\}$, $2 \to \{0,2\}$, ... (i.e., the nth row describes the n^{th} subset as follows: if there is a 1 in the k^{th} column, then k is in this subset, else it is not.) Using a similar diagonalization argument to the earlier one, flip each bit along the diagonal: $1 \to 0$, $0 \to 1$, and let b denote the resulting binary string. First, we must show that the new element is a subset of S. Clearly it is, since b is an infinite binary string which corresponds to a subset of S. Now suppose b were the n^{th} binary string. This cannot be the case though, since the n^{th} bit of b differs from the n^{th} bit of the diagonal (the bits are flipped). So it's not on our list, but it should be, since we assumed that the list enumerated all possible subsets of S. Thus we have a contradiction, implying that $\mathscr{P}(S)$ is uncountable.

Thus we have seen that the cardinality of $\mathscr{P}(\mathbb{N})$ (the power set of the natural numbers) is strictly larger than the cardinality of $\mathbb{N}$ itself. The cardinality of $\mathbb{N}$ is denoted $\aleph_0$ (pronounced "aleph null"), while that of $\mathscr{P}(\mathbb{N})$ is denoted $2^{\aleph_0}$. It turns out that in fact $\mathscr{P}(\mathbb{N})$ has the same cardinality as $\mathbb{R}$ (the real numbers), and indeed as the real numbers in $[0,1]$. This cardinality is known as $\mathbf{c}$, the "cardinality of the continuum." So we know that $2^{\aleph_0} = \mathbf{c} > \aleph_0$. Even larger infinite cardinalities (or "orders of infinity"), denoted $\aleph_1, \aleph_2, \ldots$, can be defined using the machinery of set theory; these obey (to the uninitiated somewhat bizarre) rules of arithmetic. Several fundamental questions in modern mathematics concern these objects. For example, the famous "continuum hypothesis" asserts that $\mathbf{c} = \aleph_1$ (which is equivalent to saying that there are no sets with cardinality between that of the natural numbers and that of the real numbers).

Self-Reference and Computability

In this lecture we will explore the deep connection between proofs and computation. At the heart of this connection is the notion of self-reference, and it has far-reaching consequences to the limit of computations (the Halting Problem) and the foundation of logic in mathematics (Gödel's incompleteness theorem).

The Liar's Paradox

Recall that propositions are statements that are either true or false. We saw before that some statements are not well defined or too imprecise to be called propositions. But here is a statement that is problematic for more subtle reasons:

"All Cretans are liars,"

so said a Cretan in antiquity, thus giving rise to the so-called liar's paradox which has amused and confounded people over the centuries. Why? Because if the statement above is true, then the Cretan was lying, which implies the statement is false. But actually the above statement isn't really a paradox; it simply yields a contradiction if we assume it is true, but if it is false then there is no problem.

A true formulation of this paradox is the following statement:

"This statement is false."

Is the statement above true? If the statement is true, then what it asserts must be true; namely that it is false. But if it is false, then it must be true. So it really is a paradox, and we see that it arises because of the self-referential nature of the statement. Around a century ago, this paradox found itself at the center of foundational questions about mathematics and computation.

We will now study how this paradox relates to computation. Before doing so, let us consider another manifestation of the paradox, created by the great logician Bertrand Russell. In a village with just one barber, every man keeps himself clean-shaven. Some of the men shave themselves, while others go to the barber. The barber proclaims:

"I shave all and only those men who do not shave themselves."

It seems reasonable then to ask the question: Does the barber shave himself? Thinking more carefully about the question though, we see that we are presented with the same self-referential paradox: a logically impossible scenario. If the barber does not shave himself, then according to what he announced, he shaves himself. If the barber does shave himself, then according to his statement he does not shave himself!

Self-Replicating Programs

Can we use self-reference to design a program that outputs itself? To illustrate the idea, let us consider how we can do this if we could write the program in English. Consider the following instruction:

```
Print out the following sentence:
    "Print out the following sentence:"
```

If we execute the instruction above (interpreting it as a program), then we will get the following output:

> Print out the following sentence:

Clearly this is not the same as the original instruction above, which consists of two lines. We can try to modify the instruction as follows:

```
Print out the following sentence twice:
    "Print out the following sentence twice:"
```

Executing this modified instruction yields the output which now consists of two lines:

> Print out the following sentence twice:
>
> Print out the following sentence twice:

This almost works, except that we are missing the quotes in the second line. We can fix it by modifying the instruction as follows:

```
Print out the following sentence twice, the second time in quotes:
    "Print out the following sentence twice, the second time in quotes:"
```

Then we see that when we execute this instruction, we get exactly the same output as the instruction itself:

> Print out the following sentence twice, the second time in quotes:
>
> "Print out the following sentence twice, the second time in quotes:"

Quines and the Recursion Theorem

In the above section we have seen how to write a self-replicating program in English. But can we do that in a real programming language? In general, a program that prints itself is called a *quine*,[1] and it turns out we can always write quines in any programming language.

As another example, consider the following pseudocode:

```
(Quine "s")
    (s "s")
```

The pseudocode above defines a program `Quine` that takes a string `s` as input, and outputs `(s "s")`, which means we run the string `s` (now interpreted as a program) on itself. Now consider executing the program `Quine` with input `"Quine"`:

```
(Quine "Quine")
```

By definition, this will output

> (Quine "Quine")

which is the same as the instruction that we executed!

This is a simple example, but how do we construct quines in general? The answer is given by the *recursion theorem*. The recursion theorem states that given any program $P(x,y)$, we can always convert it to another

[1] Quine is named after the philosopher and logician Willard Van Orman Quine, as popularized in the book *"Gödel, Escher, Bach: An Eternal Golden Braid"* by Douglas Hofstadter.

program $Q(x)$ such that $Q(x) = P(x, Q)$, i.e., Q behaves exactly as P would if its second input is the description of the program Q. In this sense we can think of Q as the "self-aware" version of P, since Q essentially has access to its own description.

The Halting Problem

Are there tasks that a computer cannot perform? For example, we would like to ask the following basic question when compiling a program: does it go into an infinite loop? In 1936, Alan Turing showed that there is no program that can perform this test. The proof of this remarkable fact is very elegant and combines two ingredients: self-reference (as in the liar's paradox), and the fact that we cannot separate programs from data. In computers, a program is represented by a string of bits just as integers, characters, and other data are. The only difference is in how the string of bits is interpreted.

We will now examine the Halting Problem. Given the description of a program and its input, we would like to know if the program ever halts when it is executed on the given input. In other words, we would like to write a program `TestHalt` that behaves as follows:

$$\texttt{TestHalt(P,x)} = \begin{cases} \text{“yes"}, & \text{if program } P \text{ halts on input } x \\ \text{“no"}, & \text{if program } P \text{ loops on input } x \end{cases}$$

Why can't such a program exist? First, let us use the fact that a program is just a bit string, so it can be input as data. This means that it is perfectly valid to consider the behavior of `TestHalt(P,P)`, which will output "yes" if P halts on P, and "no" if P loops forever on P. We now prove that such a program cannot exist.

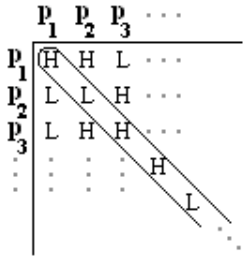
Proof: Define the program

```
Turing(P)
  if TestHalt(P,P) = "yes" then loop forever
  else halt
```

So if the program P when given P as input halts, then `Turing(P)` loops forever; otherwise, `Turing(P)` halts. Assuming we have the program `TestHalt`, we can easily use it as a subroutine in the above program `Turing`.

Now let us look at the behavior of `Turing(Turing)`. There are two cases: either it halts, or it does not. If `Turing(Turing)` halts, then it must be the case that `TestHalt(Turing,Turing)` returned "no." But by definition of `TestHalt`, that would mean that `Turing(Turing)` should not have halted. In the second case, if `Turing(Turing)` does not halt, then it must be the case that `TestHalt(Turing, Turing)` returned "yes," which would mean that `Turing(Turing)` should have halted. In both cases, we arrive at a contradiction which must mean that our initial assumption, namely that the program `TestHalt` exists, was wrong. Thus, `TestHalt` cannot exist, so it is impossible for a program to check if any general program halts. □

What proof technique did we use? This was actually a proof by diagonalization, the same technique that we used in the previous lecture to show that the real numbers are uncountable. Why? Since the set of all computer programs is countable (they are, after all, just finite-length strings over some alphabet, and the set of all finite-length strings is countable), we can enumerate all programs as follows (where P_i represents the i^{th} program):



The $(i, j)^{\text{th}}$ entry in the table above is H if program P_i halts on input P_j, and L if it does not halt. Now if the program `Turing` exists it must occur somewhere on our list of programs, say as P_n. But this cannot be, since if the n^{th} entry in the diagonal is H, meaning that P_n halts on P_n, then by its definition `Turing` loops on P_n; and if the entry is L, then by definition `Turing` halts on P_n. Thus the behavior of `Turing` is different from that of P_n, and hence `Turing` does not appear on our list. Since the list contains all possible programs, we must conclude that the program `Turing` does not exist. And since `Turing` is constructed by a simple modification of `TestHalt`, we can conclude that `TestHalt` does not exist either. Hence the Halting Problem cannot be solved.

In fact, there are many more cases of questions we would like to answer about a program, but cannot. For example, we cannot know if a program ever outputs anything or if it ever executes a specific line. We also cannot check if two programs produce the same output. We cannot even check to see if the program is a virus. These issues are explored in greater detail in the advanced course CS172 (Computability and Complexity).

The Easy Halting Problem

As noted above, the key idea in establishing the uncomputability of the Halting Problem is self-reference: Given a program P, we want to check whether $P(P)$ halts. But in practice, how often do we want to execute a program with its own description as input? Is it possible that if we disallow this kind of self-reference, we can answer the Halting Problem?

That is, given a program P, what if we ask: "Does P halts on input 0?" This looks easier than the Halting Problem (hence the name Easy Halting Problem), since we only need to check whether P halts on a specific input 0, instead of an arbitrary given input. However, it turns out this easier problem is still uncomputable. We prove this claim by showing that if we can solve the Easy Halting Problem, then we can also solve the Halting Problem; since we know the Halting Problem is uncomputable, this implies the Easy Halting Problem must also be uncomputable.

Specifically, suppose we have a program `TestEasyHalt` that answers the Easy Halting Problem:

$$\texttt{TestEasyHalt(P)} = \begin{cases} \text{"yes"}, & \text{if program } P \text{ halts on input } 0 \\ \text{"no"}, & \text{if program } P \text{ loops on input } 0 \end{cases}$$

Then we can use `TestEasyHalt` as a subroutine in the following algorithm that solves the Halting Problem:

```
Halt(P,x)
  define a program P' that returns P(x) on input 0
  return TestEasyHalt(P')
```

The algorithm `Halt` constructs another program P', which depends on both the original program P and the original input x, such that when we call $P'(0)$ we return $P(x)$. An example of such a program P' can simply be:

```
P'(y)
   return P(x)
```

That is, the new program P' ignores the input y and always returns $P(x)$. Then we see that $P'(0)$ halts if and only if $P(x)$ halts. Therefore, if we have such a program `TestEasyHalt`, then `Halt` will correctly answer the Halting Problem. Since we know there cannot be such a program `Halt`, we conclude `TestEasyHalt` does not exist either.

The technique that we use here is called a *reduction*. Here we are reducing one problem "Does P halt on x?" into another problem "Does P' halt on 0?", in the sense that if we know how to answer the second problem, then we can use that knowledge to construct an answer for the first problem. This implies that the second problem is actually as difficult as the first, despite the apparently simpler description of the second problem.

Godel's Incompleteness Theorem

In 1900, the great mathematician David Hilbert posed the following two questions about the foundation of logic in mathematics:

1. Is arithmetic consistent?
2. Is arithmetic complete?

To understand the questions above, we recall that mathematics is a formal system based on a list of axioms (for example, Peano's axioms of the natural numbers, axiom of choice, etc.) together with rules of inference. The axioms provide the initial list of true statements in our system, and we can apply the rules of inference to prove other true statements, which we can again use to prove other statements, and so on.

The first question above asks whether it is possible to prove both a proposition P and its negation $\neg P$. If this is the case, then we say that arithmetic is *inconsistent*; otherwise, we say arithmetic is *consistent*. If arithmetic is inconsistent, meaning there are false statements that can be proved, then the entire arithmetic system will collapse because from a false statement we can deduce anything, so every statement in our system will be vacuously true.

The second question above asks whether every true statement in arithmetic can be proved. If this is the case, then we say that arithmetic is *complete*. We note that given a statement, which is either true or false, it can be very difficult to prove which one it is. As a real-world example, consider the following statement, which is known as Fermat's Last Theorem:

$$\forall n \geq 3, \quad \nexists x, y, z \in \mathbb{Z}, \ x^n + y^n = z^n.$$

This theorem was first stated by Pierre de Fermat in 1637,[2] but it has eluded proofs for centuries until it was finally proved by Andrew Wiles in 1994.

In 1928, Hilbert formally posed the questions above as the Entscheidungsproblem. Most people believed that the answer would be "yes," since ideally arithmetic should be both consistent and complete. However, in 1930 Kurt Gödel proved that the answer is in fact "no": Any formal system that is sufficiently rich to formalize computation is either inconsistent (there are false statements that can be proved) or incomplete (there are true statements that cannot be proved). Gödel proved his result by exploiting the deep connection between proofs and computation, but it was not until 1936 that Turing formalized the definition of computation via the notion of Turing machines and computability.

In the rest of this note, we will first understand the essence of Gödel's proof, and then we will provide an easier proof using the Halting Problem.

[2] Along with the famous note: "I have discovered a truly marvelous proof of this, which this margin is too narrow to contain."

Sketch of Gödel's Proof

Suppose we have a formal system F, which consists of a list of axioms and rules of inference, and assume F is sufficiently expressive that we can use it to express computation. Arithmetic is an example of such a system F.

Now suppose we can write a sentence:

$$S(F) = \text{“This sentence is not provable in } F\text{.”}$$

Once we have this sentence, there are two possibilities:

1. Case 1: $S(F)$ is provable. Then the sentence $S(F)$ is true, but by inspecting the content of the sentence itself, we see that this implies $S(F)$ should not be provable. Thus, F is inconsistent in this case.
2. Case 2: $S(F)$ is not provable. By construction, this means the sentence $S(F)$ is true. Thus, F is incomplete in this case, since there is a true statement (namely, $S(F)$) that is not provable.

To complete the proof, it now suffices to construct such a sentence $S(F)$. This is the difficult part of Gödel's proof, which requires a clever encoding (Gödel numbering) of symbols and propositions as natural numbers.

Proof via the Halting Problem

Let us now see how we can prove Gödel's result by reduction to the Halting Problem. Here we proceed by contradiction: Suppose arithmetic is both consistent and complete; we will use this assumption to solve the Easy Halting Problem, which we have seen is as difficult as the Halting Problem.

Recall that in the Easy Halting Problem we want to decide whether a given program P halts on input 0. Let S denote the sentence "P halts on input 0." This is a sentence in arithmetic, and because we assume arithmetic is consistent and complete, we know that S is either true or false, and there is a proof either way.

Recall also that a proof is simply a finite binary string. How many proofs are there? There are countably many, so we can enumerate them and go through them one by one. Now define the program M as follows:

```
M(P)
  for every proof x:
    if x is a proof that P halts on 0, then output "yes"
    if x is a proof that P does not halt on 0, then output "no"
```

The program M takes as input the program P, and proceed to check every possible proof until it finds one that proves either $P(0)$ halts, or $P(0)$ does not halt. By assumption, we know there is always such a proof, so the algorithm M will terminate in finite time, and it will correctly answer the Easy Halting Problem. Since we have established that there cannot be such a program M, our initial assumption must be wrong, so it is not true that arithmetic is both consistent and complete.

Note that here we rely on the fact that given a proof, we can have a program that mechanistically check whether it is a valid proof for our proposition, and the argument above shows how we can use this to answer questions about the limit of computability. This is a manifestation of the intimate ties between proofs and computation in a deep level.

Counting

The next major topic of the course is probability theory. Suppose you toss a fair coin a thousand times. How likely is it that you get exactly 500 heads? And what about 1000 heads? It turns out that the chances of 500 heads are roughly 2.5%, whereas the chances of 1000 heads are so infinitesimally small that we may as well say that it is impossible. But before you can learn to compute or estimate odds or probabilities you must learn to count! That is the subject of this note.

We start by considering a simple scenario. We pick k elements out of an n element set $S = \{1, 2, \cdots, n\}$ one at a time. We wish to count the number of different ways to do this, taking into account the order in which the elements are picked. For example, when $k = 2$, picking 1 and then 2 is considered a different outcome from picking 2 followed by 1. Another way to ask the question is this: we wish to form an ordered sequence of k distinct elements, where each element is picked from the set S. How many different such ordered sequences are there?

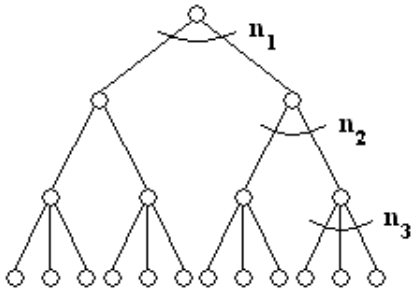
If we were dealing cards, the set would be $S = \{1, \cdots, 52\}$, where each number represents a card in a deck of 52 cards. Picking an element of S in this case refers to dealing one card. Note that once a card is dealt, it is no longer in the deck and so it cannot be dealt again. So the hand of k cards that are dealt consists of k distinct elements from the set S.

For the first card, it is easy to see that we have 52 distinct choices. But now the available choices for the second card depend upon what card we picked first. The crucial observation is that regardless of which card we picked first, there are exactly 51 choices for the second card. So the total number of ways of choosing the first two cards is 52×51. Reasoning in the same way, there are exactly 50 choices for the third card, no matter what our choices for the first two cards. It follows that there are exactly $52 \times 51 \times 50$ sequences of three cards. In general, the number of sequences of k cards is $52 \cdot 51 \cdots (52 - (k-1))$.

This is an example of the first rule of counting:

First Rule of Counting: If an object can be made by a succession of k choices, where there are n_1 ways of making the first choice, and *for every* way of making the first choice there are n_2 ways of making the second choice, and *for every* way of making the first and second choice there are n_3 ways of making the third choice, and so on up to the n_k-th choice, then the total number of distinct objects that can be made in this way is the product $n_1 \cdot n_2 \cdot n_3 \cdots n_k$.

Here is another way of picturing the first rule of counting. Consider the following tree:



It has branching factor n_1 at the root, n_2 at every node at the second level,..., n_k at every node at the k-th level. Each node at level $k+1$ (a leaf node) represents one possible way of making the object by making a succession of k choices. So the number of distinct objects that can be made is equal to the number of leaves in the tree. Moreover, the number of leaves in the tree is the product $n_1 \cdot n_2 \cdot n_3 \cdots n_k$. For example, if $n_1 = 2$, $n_2 = 2$, and $n_3 = 3$, then there are 12 leaves (i.e., outcomes).

Counting Sets

Consider a slightly different question. We would like to pick k distinct elements of $S = \{1, 2, \cdots, n\}$ (i.e. without repetition), but we do not care about the order in which we picked the k elements. For example, picking elements $1, \ldots, k$ is considered the same outcome as picking elements $2, \ldots, k$ and picking 1 as the last (k^{th} element). Now how many ways are there to choose these elements?

When dealing a hand of cards, say a poker hand, it is often more natural to count the number of distinct hands (i.e., the set of 5 cards dealt in the hand), rather than the order in which they were dealt. As we've seen in the section above, if we are considering order, there are $52 \cdot 51 \cdot 50 \cdot 49 \cdot 48 = \frac{52!}{47!}$ outcomes. But how many distinct hands of 5 cards are there? Here is another way of asking the question: each such 5 card hand is just a subset of S of cardinality 5. So we are asking how many 5 element subsets of S are there?

Here is a clever trick for counting the number of distinct subsets of S with exactly 5 elements. Create a bin corresponding to each such 5 element subset. Now take all the sequences of 5 cards and distribute them into these bins in the natural way. Each sequence gets placed in the bin corresponding to the set of 5 elements in the sequence. Thus if the sequence is $(2, 1, 3, 5, 4)$, then it is placed in the bin labeled $\{1, 2, 3, 4, 5\}$. How many sequences are placed in each bin? The answer is exactly 5!, since there are exactly 5! different ways to order 5 cards.

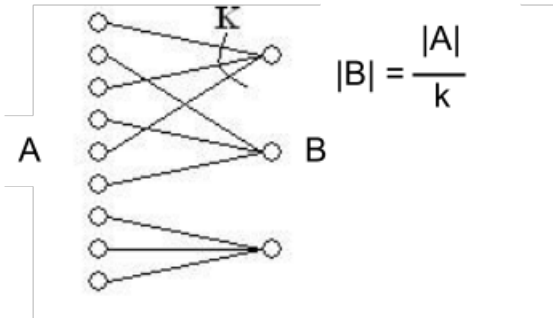
Recall that our goal was to compute the number of 5 element subsets, which now corresponds to the number of bins. We know that there are $\frac{52!}{47!}$ 5-card sequences, and there are 5! sequences placed in each bin. The total number of bins is therefore $\frac{52!}{47!5!}$.

This quantity $\frac{n!}{(n-k)!k!}$ is used so often that there is special notation for it: $\binom{n}{k}$, pronounced *n choose k*. This is the number of ways of picking k distinct elements from S, where the order of placement does not matter. Equivalently, it's the number of ways of choosing k objects out of a total of n objects, where the order of the choices does not matter.

The trick we used above is actually our second rule of counting:

Second Rule of Counting: Assume an object is made by a succession of choices, and the order in which the choices is made does not matter. Let A be the set of ordered objects and let B be the set of unordered objects. If there exists a k to 1 function f from A to B, we can count the number of ordered objects (pretending that the order matters) and divide by k (the number of ordered objects per unordered objects) to obtain $|B|$, the number of unordered objects.

Note that we are assuming the number of ordered objects is the same for every unordered object; the rule cannot be applied otherwise. Here is another way of picturing the second rule of counting:



The function f simply places the ordered outcomes into bins corresponding to the unordered outcomes. In our poker hand example, f will map 5! elements in the domain of the function (the set of ordered 5 card outcomes) to one element in the range (the set of 5 element subsets of S). The number of elements in the range of the function is therefore $\frac{52!}{47!5!}$.

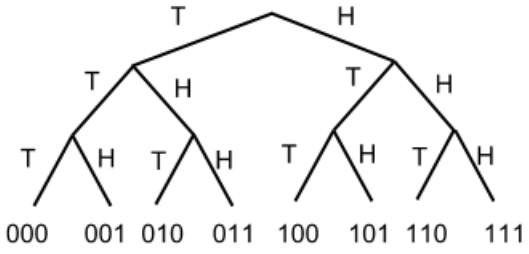
Sampling with Replacement

Sometimes we wish to consider a different scenario where we are still picking k elements out of an n element set $S = \{1, 2, \cdots, n\}$ one at a time (order matters). The difference is that after we pick an element for our sequence, we throw it back into S so we can pick it again. How many such sequences of k elements can we obtain? We can use the first rule of counting. Since we have n choices in each trial, $n_1 = n_2 = \ldots = n_k = n$. Then we have a grand total of n^k sequences.

This type of sampling is called *sampling with replacement*; multiple trials can have the same outcome. Card dealing is a type of *sampling without replacement*, since two trials cannot both have the same outcome (one card cannot be dealt twice).

Coin Flipping

Let us return to coin flipping. How many different outcomes are there if we flip a coin k times? By outcome, we mean the string of results: i.e. 001 would represent 2 tails followed by a heads. We can picture this using the tree below, which depicts the possible outcomes when $k = 3$:



Here $S = \{0, 1\}$. This is a case of sampling with replacement; multiple coin flips could result in tails (we could pick the element 0 from set S in multiple trials). Order also matters - strings 001 and 010 are

considered different outcomes. By the reasoning above (using the first rule of counting) we have a total of $n^k = 2^k$ distinct outcomes, since $n = 2$.

Rolling Dice

Let's say we roll two dice, so $k = 2$ and $S = \{1,2,3,4,5,6\}$. How many possible outcomes are there? In this setting, ordering matters; obtaining 1 with the first die and 2 with the second is different from obtaining 2 with the first and 1 with the second. We are sampling with replacement, since we can obtain the same result on both dice.

The setting is therefore the same as the coin flipping example above (order matters and we are sampling with replacement), so we can use the first rule of counting in the same manner. The number of distinct outcomes is therefore $n^2 = 6^2 = 36$.

Sampling with replacement, but where order does not matter

Say you have unlimited quantities of apples, bananas and oranges. You want to select 5 pieces of fruit to make a fruit salad. How many ways are there to do this? In this example, $S = \{1,2,3\}$, where 1 represents apples, 2 represents bananas, and 3 represents oranges. $k = 5$ since we wish to select 5 pieces of fruit. Ordering does not matter; selecting an apple followed by a banana will lead to the same salad as a banana followed by an apple.

This scenario is much more tricky to analyze. It is natural to apply the second rule of counting because order does not matter. So we first pretend that order matters, and then the number of ordered objects is 3^5 as discussed above. How many ordered options are there for every unordered option? The problem is that this number differs depending on which unordered object we are considering. Let's say the unordered object is an outcome with 5 bananas. There is only one such ordered outcome. But if we are considering 4 bananas and 1 apple, there are 5 such ordered outcomes (represented as 12222,21222,22122,22212,22221).

Now that we see the second rule of counting will not help, can we look at this problem in a different way? Let us first generalize back to our original setting: we have a set $S = \{1,2,\cdots,n\}$ and we would like to know how many ways there are to choose multisets (sets with repetition) of size k. Remarkably, we can model this problem in terms of binary strings.

Assume we have 1 bin for each element from set S, so n bins. For example, if we selected 2 apples and 1 banana, bin 1 would have 2 elements and bin 2 would have 1 element. In order to count the number of multisets, we need to count how many different ways there are to fill these bins with k elements. We don't care about the order of the bins themselves, just how many of the k elements each bin contains. Let's represent each of the k elements by a 0 in the binary string, and separations between bins by a 1. Consider the following picture:

This would be a sample placement where $S = \{1,\cdots,5\}$ and $k = 4$. Counting the number of multi sets is now equivalent to counting the number of placements of the k 0's. We have just reduced what seemed like a very complex problem to a question about a binary string, simply by looking at it from a different perspective!

How many ways can we choose these locations? The length of our binary string is $k+n-1$, and we are

choosing which k locations should contain 0's. The remaining $n-1$ locations will contain 1's. Once we pick a location for one zero, we cannot pick it again; repetition is not allowed. Picking location 1 followed by location 2 is the same as picking location 2 followed by location 1, so ordering does not matter. It follows that all we wish to compute is the number of ways of picking k elements from $k+n-1$ elements, without replacement and where the order of placement does not matter. This is given by $\binom{n+k-1}{k}$, as discussed in the Counting Sets section above. This is therefore the number of ways in which we can choose multisets of size k from set S.

Returning to our example above, the number of ways of picking 5 pieces of fruit is exactly $\binom{3+5-1}{5} = \binom{7}{5}$

Notice that we started with a problem which seemed very different from previous examples, but, by viewing it from a certain perspective, we were able to use previous techniques (those used in counting sets) to find a solution! This is key to many combinatorial arguments as we will explore further in the next section.

Combinatorial Proofs

Combinatorial arguments are interesting because they rely on intuitive counting arguments rather than algebraic manipulation. We can prove complex facts, such as $\binom{n}{k+1} = \binom{n-1}{k} + \binom{n-2}{k} + \cdots + \binom{k}{k}$. You can directly verify this identity by algebraic manipulation. But you can also do this by interpreting what each side means as a combinatorial process. The left hand side is just the number of ways of choosing a $k+1$ element subset from a set of n items. Let us think about a different process that results in the choice of a $k+1$ element subset. We start by picking the lowest numbered element in the subset. It is either the first element of the set or the second or the third or ... If we choose the first element, we must now choose k elements out of the remaining $n-1$ which we can do in $\binom{n-1}{k}$ ways. If instead the lowest numbered element we picked is the second element then we still have to choose k elements from the remaining $n-2$ which can be done in $\binom{n-2}{k}$ ways. Moreover all these subsets are distinct from those where the lowest numbered element was the first one. So we should add the number of ways of choosing each to the grand total. Proceeding in this way, we split up the process into cases according to the first (i.e., lowest-numbered) object we select, to obtain:

$$\textbf{First element selected is either} \begin{cases} \text{element } 1, & \binom{n-1}{k} \\ \text{element } 2, & \binom{n-2}{k} \\ \text{element } 3, & \binom{n-3}{k} \\ \vdots & \\ \text{element}(n-k), & \binom{k}{k} \end{cases}$$

(Note that the lowest-numbered object we select cannot be higher than $n-k$ as we have to select k distinct objects.)

The last combinatorial proof we will do is the following: $\binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{n} = 2^n$. To see this, imagine that we have a set S with n elements. On the left hand side, the i^{th} term counts the number of ways of choosing a subset of S of size exactly i; so the sum on the left hand side counts the total number of subsets (of any size) of S.

We claim that the right hand side (2^n) does indeed also count the total number of subsets. To see this, just identify a subset with an n-bit vector, where in each position j we put a 1 if the jth element is in the subset, and a 0 otherwise. So the number of subsets is equal to the number of n-bit vectors, which is 2^n (there are 2 options for each bit). Let us look at an example, where $S = \{1,2,3\}$ (so $n = 3$). Enumerate all $2^3 = 8$ possible subsets of S: $\{\{\},\{1\},\{2\},\{3\},\{1,2\},\{1,3\},\{2,3\},\{1,2,3\}\}$. The term $\binom{3}{0}$ counts the number of

ways to choose a subset of S with 0 elements; there is only one such subset, namely the empty set. There are $\binom{3}{1} = 3$ ways of choosing a subset with 1 element, $\binom{3}{2} = 3$ ways of choosing a subset with 2 elements, and $\binom{3}{3} = 1$ way of choosing a subset with 3 elements (namely, the subset consisting of the whole of S). Summing, we get $1+3+3+1 = 8$, as expected.

Introduction to Discrete Probability

In the last note we considered the probabilistic experiment where we flipped a fair coin $10,000$ times and counted the number of Hs. We asked "what is the chance that we get between 4900 and 5100 Hs". One of the lessons was that the remarkable concentration of the fraction of Hs had something to do with the astronomically large number of possible outcomes of $10,000$ coin flips.

In this note we will formalize all these notions for an arbitrary probabilistic experiment. We will start by introducing the space of all possible outcomes of the experiment, called a sample space. Each element of the sample space is assigned a probability - which tells us how likely it is to occur when we actually perform the experiment. The mathematical formalism we introduce might take you some time to get used to. But you should remember that ultimately it is just a precise way to say what we mean when we describe a probabilistic experiment like flipping a coin n times.

Random Experiments

In general, a probabilistic experiment consists of drawing a sample of k elements from a set S of cardinality n. The possible outcomes of such an experiment are exactly the objects that we counted in the last note. Recall from the last note that we considered four possible scenarios for counting, depending upon whether we sampled with or without replacement, and whether the order in which the k elements are chosen does or does not matter. The same will be the case for our probabilistic experiments. The outcome of the random experiment is called a *sample point*. The *sample space*, often denoted by Ω, is the set of all possible outcomes.

An example of such an experiment is tossing a coin 4 times. In this case, $S = \{H, T\}$ and we are drawing 4 elements with replacement. $HTHT$ is an example of a sample point and the sample space has 16 elements:

$\Omega =$

HHHH HTHH
HHHT HTHT
HHTH HTTH
HHTT HTTT
THHH TTHH
THHT TTHT
THTH TTTH
THTT TTTT

How do we determine the chance of each particular outcome, such as HHT, of our experiment? In order to do this, we need to define the probability for each sample point, as we will do below.

Probability Spaces

A probability space is a sample space Ω, together with a probability $\Pr[\omega]$ for each sample point ω, such that

- $0 \leq \Pr[\omega] \leq 1$ for all $\omega \in \Omega$.
- $\sum_{\omega \in \Omega} \Pr[\omega] = 1$, i.e., the sum of the probabilities of all outcomes is 1.

The easiest way to assign probabilities to sample points is uniformly: if $|\Omega| = N$, then $\Pr[x] = \frac{1}{N}\ \forall x \in \Omega$. For example, if we toss a fair coin 4 times, each of the 16 sample points (as pictured above) is assigned probability $\frac{1}{16}$. We will see examples of non-uniform probability distributions soon.

After performing an experiment, we are often interested in knowing whether an event occurred. For example, we might be interested in the event that there were "exactly 2 H's in four tosses of the coin". How do we formally define the concept of an event in terms of the sample space Ω? Here is a beautiful answer. We will identify the event "exactly 2 H's in four tosses of the coin" with the subset consisting of those outcomes in which there are exactly two H's:
$\{HHTT, HTHT, HTTH, THHT, THTH, TTHH\} \subseteq \Omega$. Now we turn this around and say that formally an event A is just a subset of the sample space, $A \subseteq \Omega$.

How should we define the probability of an event A? Naturally, we should just *add up* the probabilities of the sample points in A.

For any event $A \subseteq \Omega$, we define the probability of A to be

$$\Pr[A] = \sum_{\omega \in A} \Pr[\omega].$$

Thus the probability of getting exactly two H's in four coin tosses can be calculated using this definition as follows. A consists of all sequences that have exactly two H's, and so $|A| = \binom{4}{2} = 6$. For this example, there are $2^4 = 16$ possible outcomes for flipping four coins. Thus, each sample point $\omega \in A$ has probability $\frac{1}{16}$; and, as we saw above, there are six sample points in A, giving us $\Pr[A] = 6 \cdot \frac{1}{16} = \frac{3}{8}$.

Examples

We will now look at examples of random experiments and their corresponding sample spaces, along with possible probability spaces and events.

Coin Flipping

Suppose we have a coin of bias p, and our experiment consists of flipping the coin 4 times. The sample space Ω consists of the sixteen possible sequences of H's and T's shown in the figure on the last page.

The probability space depends on p. If $p = \frac{1}{2}$ the probabilities are assigned uniformly; the probability of each sample point is $\frac{1}{16}$. What if the coin comes up heads with probability $\frac{2}{3}$ and tails with probability $\frac{1}{3}$ (i.e. the bias is $p = \frac{2}{3}$)? Then the probabilities are different. For example, $\Pr[HHHH] = \frac{2}{3} \times \frac{2}{3} \times \frac{2}{3} \times \frac{2}{3} = \frac{16}{81}$, while $\Pr[TTHH] = \frac{1}{3} \times \frac{1}{3} \times \frac{2}{3} \times \frac{2}{3} = \frac{4}{81}$. [*Note: We have cheerfully* multiplied *probabilities here; we'll explain why this is OK later. It is* not *always OK!*]

What type of events can we consider in this setting? Let event A be the event that all four coin tosses are the same. Then $A = \{HHHH, TTTT\}$. $HHHH$ has probability $\frac{2}{3}^4$ and $TTTT$ has probability $\frac{1}{3}^4$. Thus, $\Pr[A] = \Pr[HHHH] + \Pr[TTTT] = \frac{2}{3}^4 + \frac{1}{3}^4 = \frac{17}{81}$.

Next, consider event B: the event that there are exactly two heads. The probability of any particular outcome with two heads (such as $HTHT$) is $\frac{2}{3}^2\frac{1}{3}^2$. How many such outcomes are there? There are $\binom{4}{2} = 6$ ways of choosing the positions of the heads, and these choices completely specify the sequence. So $\Pr[B] = 6\frac{2}{3}^2\frac{1}{3}^2 = \frac{24}{81} = \frac{8}{27}$.

More generally, if we flip the coin n times, we get a sample space Ω of cardinality 2^n. The sample points are all possible sequences of n H's and T's. If the coin has bias p, and if we consider any sequence of n coin flips with exactly r H's, then the the probability of this sequence is $p^r(1-p)^{n-r}$.

Now consider the event C that we get exactly r H's when we flip the coin n times. This event consists of exactly $\binom{n}{r}$ sample points. Each has probability $p^r(1-p)^{n-r}$. So the probability of this event, $P[C] = \binom{n}{r}p^r(1-p)^{n-r}$.

Biased coin-tossing sequences show up in many contexts: for example, they might model the behavior of n trials of a faulty system, which fails each time with probability p.

Rolling Dice

The next random experiment we will discuss consists of rolling two dice. In this experiment, $\Omega = \{(i, j) : 1 \leq i, j \leq 6\}$. The probability space is uniform, i.e. all of the sample points have the *same* probability, which must be $\frac{1}{|\Omega|}$. In this case, $|\Omega| = 36$, so each sample point has probability $\frac{1}{36}$. In such circumstances, the probability of any event A is clearly just

$$\Pr[A] = \frac{\text{\# of sample points in } A}{\text{\# of sample points in } \Omega} = \frac{|A|}{|\Omega|}.$$

So for uniform spaces, computing probabilities reduces to *counting* sample points!

Now consider two events: the event A that the sum of the dice is at least 10 and the event B that there is at least one 6. By writing out the number of sample points in each event, we can determine the number of sample points in each event; $|A| = 6$ and $|B| = 11$. By the observation above, it follows that $\Pr[A] = \frac{6}{36} = \frac{1}{6}$ and $\Pr[B] = \frac{11}{36}$.

Card Shuffling

The random experiment consists of shuffling a deck of cards. Ω is equal to the set of the 52! permutations of the deck. The probability space is uniform. Note that we're really talking about an idealized mathematical model of shuffling here; in real life, there will always be a bit of bias in our shuffling. However, the mathematical model is close enough to be useful.

Poker Hands

Here's another experiment: shuffling a deck of cards and dealing a poker hand. In this case, S is the set of 52 cards and our sample space $\Omega = \{\text{all possible poker hands}\}$, which corresponds to choosing $k = 5$ objects without replacement from a set of size $n = 52$ where order does not matter. Hence, as we saw in the previous Note, $|\Omega| = \binom{52}{5} = \frac{52\times51\times50\times49\times48}{5\times4\times3\times2\times1} = 2,598,960$. Since the deck is assumed to be randomly shuffled, the probability of each outcome is equally likely and we are therefore dealing with a uniform probability space.

Let A be the event that the poker hand is a flush. [For those who are not addicted to gambling, a *flush* is a hand in which all cards have the same suit, say Hearts.] Since the probability space is uniform, computing $\Pr[A]$ reduces to simply computing $|A|$, or the number of poker hands which are flushes. There are 13 cards in each suit, so the number of flushes in each suit is $\binom{13}{5}$. The total number of flushes is therefore $4 \cdot \binom{13}{5}$. Then we have

$$\Pr[\text{hand is a flush}] = \frac{4 \cdot \binom{13}{5}}{\binom{52}{5}} = \frac{4 \cdot 13! \cdot 5! \cdot 47!}{5! \cdot 8! \cdot 52!} = \frac{4 \cdot 13 \cdot 12 \cdot 11 \cdot 10 \cdot 9}{52 \cdot 51 \cdot 50 \cdot 49 \cdot 48} \approx 0.002.$$

Balls and Bins

In this experiment, we will throw 20 (labeled) balls into 10 (labeled) bins. Assume that each ball is equally likely to land in any bin, regardless of what happens to the other balls.

If you wish to understand this situation in terms of sampling a sequence of k elements from a set S of cardinality n: here the set S consists of the 10 bins, and we are sampling with replacement $k = 20$ times. The order of sampling matters, since the balls are labeled.

The sample space Ω is equal to $\{(b_1, b_2, \ldots, b_{20}) : 1 \leq b_i \leq 10\}$, where the component b_i denotes the bin in which ball i lands. The cardinality of the sample space, $|\Omega|$, is equal to 10^{20}- each element b_i in the sequence has 10 possible choices, and there are 20 elements in the sequence. More generally, if we throw m balls into n bins, we have a sample space of size n^m. The probability space is uniform; as we said earlier, each ball is equally likely to land in any bin.

Let A be the event that bin 1 is empty. Since the probability space is uniform, we simply need to count how many outcomes have this property. This is exactly the number of ways all 20 balls can fall into the remaining nine bins, which is 9^{20}. Hence, $\Pr[A] = \frac{9^{20}}{10^{20}} = (\frac{9}{10})^{20} \approx 0.12$.

Let B be the event that bin 1 contains at least one ball. This event is the *complement* $\bar{A}$ of A, i.e., it consists of precisely those sample points which are not in A. So $\Pr[B] = 1 - \Pr[A] \approx .88$. More generally, if we throw m balls into n bins, we have:

$$\Pr[\text{bin 1 is empty}] = \left(\frac{n-1}{n}\right)^m = \left(1 - \frac{1}{n}\right)^m.$$

As we shall see, balls and bins is another probability space that shows up very often in Computer Science: for example, we can think of it as modeling a load balancing scheme, in which each job is sent to a random processor.

It is also a more general model for problems we have previously considered. For example, flipping a fair coin 3 times is a special case in which the number of balls (m) is 3 and the number of bins (n) is 2. Rolling two dice is a special case in which $m = 2$ and $n = 6$.

Birthday Paradox

The "birthday paradox" is a remarkable phenomenon that examines the chances that two people in a group have the same birthday. It is a "paradox" not because of a logical contradiction, but because it goes against intuition. For ease of calculation, we take the number of days in a year to be 365. Then $U = \{1, \ldots, 365\}$, and the random experiment consists of drawing a sample of n elements from U, where the elements are the birth dates of n people in a group. Then $|\Omega| = 365^n$. This is because each sample point is a sequence of possible birthdays for n people; so there are n points in the sequence and each point has 365 possible values.

Let A be the event that at least two people have the same birthday. If we want to determine $\Pr[A]$, it might be simpler to instead compute the probability of the complement of A, $\Pr[\bar{A}]$. $\bar{A}$ is the event that no two people have the same birthday. Since $\Pr[A] = 1 - \Pr[\bar{A}]$, we can then easily compute $\Pr[A]$.

We are again working in a uniform probability space, so we just need to determine $|\bar{A}|$. Equivalently, we are computing the number of ways there are for no two people to have the same birthday. There are 365 choices for the first person, 364 for the second, $\ldots$, $365 - n + 1$ choices for the n^{th} person, for a total of $365 \times 364 \times \cdots \times (365 - n + 1)$. Note that this is simply an application of the first rule of counting; we are sampling without replacement and the order matters.

Thus we have $\Pr[\bar{A}] = \frac{|\bar{A}|}{|\Omega|} = \frac{365 \times 364 \times \cdots \times (365-n+1)}{365^n}$. Then $\Pr[A] = 1 - \frac{365 \times 364 \times \cdots \times (365-n+1)}{365^n}$. This allows us to compute $\Pr[A]$ as a function of the number of people, n. Of course, as n increases $\Pr[A]$ increases. In fact, with $n = 23$ people you should be willing to bet that at least two people do have the same birthday, since then $\Pr[A]$ is larger than 50%! For $n = 60$ people, $\Pr[A]$ is over 99%.

The Monty Hall Problem

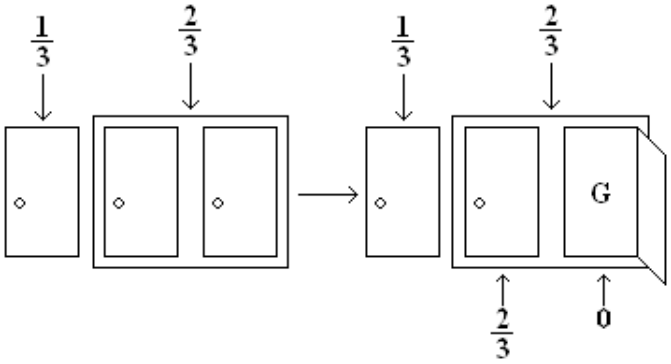
In an (in)famous 1970s game show hosted by one Monty Hall, a contestant was shown three doors; behind one of the doors was a prize, and behind the other two were goats. The contestant picks a door (but doesn't open it). Then Hall's assistant (Carol), opens one of the other two doors, revealing a goat (since Carol knows where the prize is, she can always do this). The contestant is then given the option of sticking with his current door, or switching to the other unopened one. He wins the prize if and only if his chosen door is the correct one. The question, of course, is: Does the contestant have a better chance of winning if he switches doors?

Intuitively, it seems obvious that since there are only two remaining doors after the host opens one, they must have equal probability. So you may be tempted to jump to the conclusion that it should not matter whether or not the contestant stays or switches.

Yet there are other people whose intuition cries out that the contestant is better off switching. So who's correct?

As a matter of fact, the contestant has a better chance of picking the car if he uses the switching strategy. How can you convince yourself that this is true? One way you can do this is by doing a rigorous analysis. You would start by writing out the sample space, and then assign probabilities to each sample point. Finally you would calculate the probability of the event that the contestant wins under the sticking strategy. This is an excellent exercise if you wish to make sure you understand the formalism of probability theory we introduced above.

Let us instead give a more intuitive pictorial argument. Initially when the contestant chooses the door, he has a $\frac{1}{3}$ chance of picking the car. This must mean that the other doors combined have a $\frac{2}{3}$ chance of winning. But after Carol opens a door with a goat behind it, how do the probabilities change? Well, everyone knows that there is a goat behind one of the doors that the contestant did not pick. So no matter whether the contestant is winning or not, Carol is always able to open one of the other doors to reveal a goat. This means that the contestant still has a $\frac{1}{3}$ chance of winning. Also the door that Carol opened has no chance of winning. What about the last door? It must have a $\frac{2}{3}$ chance of containing the car, and so the contestant has a higher chance of winning if he or she switches doors. This argument can be summed up nicely in the following picture:



You will be able to formalize this intuitive argument once we cover conditional probability.

In the meantime, to approach this problem formally, first determine the sample space and the probability space. Just a hint: it is not a uniform probability space! Then formalize the event we have described above (as a subspace of the sample space), and compute the probability of the event. Good luck!

Summary

The examples above illustrate the importance of doing probability calculations systematically, rather than "intuitively." Recall the key steps in all our calculations:

- What is the sample space (i.e., the experiment and its set of possible outcomes)?
- What is the probability of each outcome (sample point)?
- What is the event we are interested in (i.e., which subset of the sample space)?
- Finally, compute the probability of the event by adding up the probabilities of the sample points inside it.

Whenever you meet a probability problem, you should always go back to these basics to avoid potential pitfalls. Even experienced researchers make mistakes when they forget to do this — witness many erroneous "proofs", submitted by mathematicians to newspapers at the time, of the fact that the switching strategy in the Monty Hall problem does not improve the odds.

Introduction

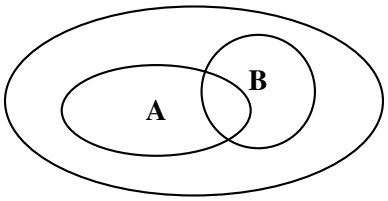
One of the key properties of coin flips is independence: if you flip a fair coin ten times and get ten T's, this does not make it more likely that the next coin flip will be H's. It still has exactly 50% chance of being H's.

By contrast, suppose while dealing cards, the first ten cards are all red (hearts or diamonds). What is the chance that the next card is red? This is easy: we started with exactly 26 red cards and 26 black cards. But after dealing the first ten cards we know that the deck has 16 red cards and 26 black cards. So the chance that the next card is red is $\frac{16}{42}$. So unlike the case of coin flips, now the chance of drawing a red card is no longer independent of the previous card that was dealt. This is the phenomenon we will explore in this note on conditional probability.

Conditional Probability

Let's consider an example with a smaller sample space. Suppose we toss $m = 3$ balls into $n = 3$ bins; this is a uniform sample space with $3^3 = 27$ points. We already know that the probability the first bin is empty is $(1 - \frac{1}{3})^3 = (\frac{2}{3})^3 = \frac{8}{27}$. What is the probability of this event *given that* the second bin is empty? Call these events A, B respectively. In the language of conditional probability we wish to compute the probability $P[A|B]$, which we read to say probability of A given B.

How should we compute $\Pr[A|B]$? Well, since event B is guaranteed to happen, we need to look not at the whole sample space Ω, but at the smaller sample space consisting only of the sample points in B. In terms of the picture below, we are no longer looking at the large oval, but only the oval labeled B:



What should the probabilities of these sample points be? If they all simply inherit their probabilities from Ω, then the sum of these probabilities will be $\sum_{\omega \in B} \Pr[\omega] = \Pr[B]$, which in general is less than 1. So we need to *scale* the probability of each sample point by $\frac{1}{\Pr[B]}$. I.e., for each sample point $\omega \in B$, the new probability becomes

$$\Pr[\omega|B] = \frac{\Pr[\omega]}{\Pr[B]}.$$

Now it is clear how to compute $\Pr[A|B]$: namely, we just sum up these scaled probabilities over all sample points that lie in both A and B:

$$\Pr[A|B] = \sum_{\omega \in A \cap B} \Pr[\omega|B] = \sum_{\omega \in A \cap B} \frac{\Pr[\omega]}{\Pr[B]} = \frac{\Pr[A \cap B]}{\Pr[B]}.$$

Definition 14.1 (Conditional Probability)**.** *For events A, B in the same probability space, such that* $\Pr[B] > 0$*, the conditional probability of A given B is*

$$\Pr[A|B] = \frac{\Pr[A \cap B]}{\Pr[B]}.$$

Returning to our example, to compute $\Pr[A|B]$ we need to figure out $\Pr[A \cap B]$. But $A \cap B$ is the event that both the first two bins are empty, i.e., all three balls fall in the third bin. So $\Pr[A \cap B] = \frac{1}{27}$ (why?). Therefore,

$$\Pr[A|B] = \frac{\Pr[A \cap B]}{\Pr[B]} = \frac{1/27}{8/27} = \frac{1}{8}.$$

Not surprisingly, $\frac{1}{8}$ is quite a bit less than $\frac{8}{27}$: knowing that bin 2 is empty makes it significantly less likely that bin 1 will be empty.

Example: Card Dealing

Let's apply the ideas discussed above to compute the probability that, when dealing 2 cards and the first card is known to be an ace, the second card is also an ace. Let B be the event that the first card is an ace, and let A be the event that the second card is an ace. Note that $P[A] = P[B] = \frac{1}{13}$.

To compute $\Pr[A|B]$, we need to figure out $\Pr[A \cap B]$. This is the probability that both cards are aces. Note that there are $52 \cdot 51$ sample points in the sample space, since each sample point is a sequence of two cards. A sample point is in $A \cap B$ if both cards are aces. This can happen in $4 \cdot 3 = 12$ ways.

Since each sample point is equally likely, $\Pr[A \cap B] = \frac{12}{52 \cdot 51}$. The probability of event B, drawing an ace in the first trial, is $\frac{4}{52}$. Therefore,

$$\Pr[A|B] = \frac{\Pr[A \cap B]}{\Pr[B]} = \frac{3}{51}.$$

Note that this says that if the first card is an ace, it makes it less likely that the second card is also an ace.

Bayesian Inference

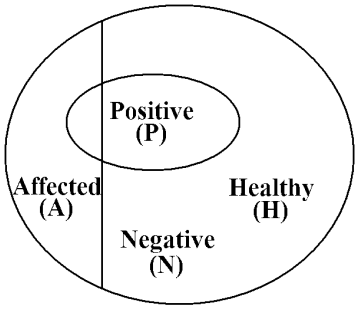
Now that we've introduced the notion of conditional probability, we can see how it is used in real world settings. Conditional probability is at the heart of a subject called *Bayesian inference*, used extensively in fields such as machine learning, communications and signal processing. Bayesian inference is a way to *update knowledge* after making an observation. For example, we may have an estimate of the probability of a given event A. After event B occurs, we can update this estimate to $\Pr[A|B]$. In this interpretation, $\Pr[A]$ can be thought of as a *prior* probability: our assessment of the likelihood of an event of interest A *before* making an observation. It reflects our prior knowledge. $\Pr[A|B]$ can be interpreted as the *posterior* probability of A after the observation. It reflects our new knowledge.

Here is an example of where we can apply such a technique. A pharmaceutical company is marketing a new test for a certain medical disorder. According to clinical trials, the test has the following properties:

1. When applied to an affected person, the test comes up positive in 90% of cases, and negative in 10% (these are called "false negatives").

2. When applied to a healthy person, the test comes up negative in 80% of cases, and positive in 20% (these are called "false positives").

Suppose that the incidence of the disorder in the US population is 5%; this is our prior knowledge. When a random person is tested and the test comes up positive, how can we update this probability? (Note that this is presumably *not* the same as the simple probability that a random person has the disorder, which is just $\frac{1}{20}$.) The implicit probability space here is the entire US population with uniform probabilities.



The sample space here consists of all people in the US — denote their number by N (so $N \approx 250$ million). Let A be the event that a person chosen at random is affected, and B be the event that a person chosen at random tests positive. Now we can rewrite the information above:

- $\Pr[A] = 0.05$, (5% of the U.S. population is affected)
- $\Pr[B|A] = 0.9$ (90% of the affected people test positive)
- $\Pr[B|\overline{A}] = 0.2$ (20% of healthy people test positive)

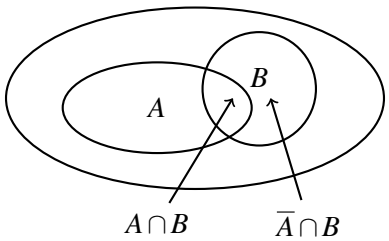
We want to calculate $\Pr[A|B]$. We can proceed as follows:

$$\Pr[A|B] = \frac{\Pr[A \cap B]}{\Pr[B]} = \frac{\Pr[B|A]\Pr[A]}{\Pr[B]} \tag{1}$$

We obtained the second equality above by applying the definition of conditional probability:

$$\Pr[B|A] = \frac{\Pr[A \cap B]}{\Pr[A]}$$

Now we need to compute $\Pr[B]$. This is the probability that a random person tests positive. To compute this, we can sum two values: the probability that a healthy person tests positive, $\Pr[\bar{A} \cap B]$ and the probability that an affected person tests positive, $\Pr[A \cap B]$. We can sum because the events $\bar{A} \cap B$ and $A \cap B$ do not intersect:



By again applying the definition of conditional probability we have:

$$\Pr[B] = \Pr[A \cap B] + \Pr[\overline{A} \cap B] = \Pr[B|A]\Pr[A] + \Pr[B|\overline{A}](1 - \Pr[A]) \tag{2}$$

Combining equations (1) and (2), we have expressed $\Pr[A|B]$ in terms of $\Pr[A], \Pr[B|A]$ and $\Pr[B|\overline{A}]$:

$$\Pr[A|B] = \frac{\Pr[B|A]\Pr[A]}{\Pr[B|A]\Pr[A] + \Pr[B|\overline{A}](1-\Pr[A])} \tag{3}$$

By plugging in the values written above, we obtain $\Pr[A|B] = \frac{9}{47} \approx .19$.

Equation (3) is useful for many inference problems. We are given $\Pr[A]$, which is the (unconditional) probability that the event of interest A happens. We are given $\Pr[B|A]$ and $\Pr[B|\overline{A}]$, which quantify how noisy the observation is. (If $\Pr[B|A] = 1$ and $\Pr[B|\overline{A}] = 0$, for example, the observation is completely noiseless.) Now we want to calculate $\Pr[A|B]$, the probability that the event of interest happens given we made the observation. Equation (3) allows us to do just that.

Of course, equations (1), (2) and (3) are derived from the basic axioms of probability and the definition of conditional probability, and are therefore true with or without the above Bayesian inference interpretation. However, this interpretation is very useful when we apply probability theory to study inference problems.

Bayes' Rule and Total Probability Rule

Equations (1) and (2) are very useful in their own right. The first is called **Bayes' Rule** and the second is called the **Total Probability Rule**. Bayes' rule is useful when one wants to calculate $\Pr[A|B]$ but one is given $\Pr[B|A]$ instead, i.e. it allows us to "flip" things around.

The Total Probability rule is an application of the strategy of "dividing into cases" .There are two possibilities: either an event A happens or A does not happen. If A happens the probability that B happens is $\Pr[B|A]$. If A does not happen, the probability that B happens is $\Pr[B|\overline{A}]$. If we know or can easily calculate these two probabilities and also $\Pr[A]$, then the total probability rule yields the probability of event B.

Example: Tennis Match

You are about to play a tennis match against a randomly chosen opponent and you wish to calculate your probability of winning. You know your opponent will be one of two people, X or Y. If person X is chosen, you will win with probability .7. If person Y is chosen, you will win with probability .3. Your opponent is chosen by flipping a coin with bias .6 in favor of X.

Let's first determine which events we are interested in. Let A be the event that you win. Let B_1 be the event that person X is chosen, and let B_2 be the event that person Y is chosen. We wish to calculate $\Pr[A]$. Here is what we know so far:

- $\Pr[A|B_1] = 0.7$, (if person X is chosen, you win with probability .7)
- $\Pr[A|B_2] = 0.3$ (if person Y is chosen, you win with probability .3)
- $\Pr[B_1] = 0.6$ (person X is chosen with probability .6)
- $\Pr[B_2] = 0.4$ (person Y is chosen with probability .4)

By using the Total Probability rule, we have:

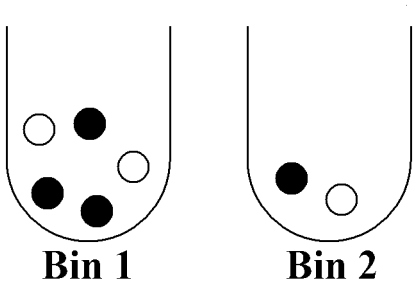
$$\Pr[A] = \Pr[A|B_1]\Pr[B_1] + \Pr[A|B_2]\Pr[B_2].$$

Now we can simply plug in the known values above to obtain $\Pr[A]$:

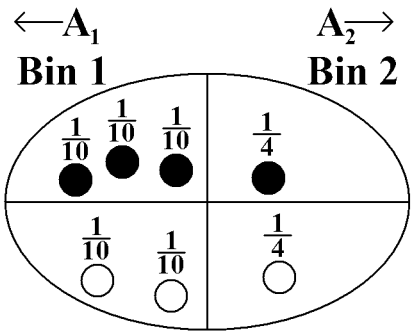
$$\Pr[A] = .7 \times .6 + .3 \times .4 = .54$$

Example: Balls and Bins

Imagine we have two bins containing black and white balls, and further suppose that we wanted to know what is the chance that we picked Bin 1 given that we picked a white ball, i.e., $\Pr[\text{Bin } 1|\bigcirc]$. Assume that we are unbiased when choosing a bin so that each bin is chosen with probability $\frac{1}{2}$.



A wrong approach is to say that the answer is clearly $\frac{2}{3}$, since we know there are a total of three white balls, two of which are in bin 1. However, this picture is misleading because the bins have equal "weight". Instead, what we should do is appropriately scale each sample point as the following picture shows:



This images shows that the sample space Ω is equal to the union of the events contained in bin $1(A_1)$ and bin $2(A_2)$, so $\Omega = A_1 \cup A_2$. We can use the definition of conditional probability to see that

$$\Pr[\text{Bin } 1|\bigcirc] = \frac{\frac{1}{10}+\frac{1}{10}}{\frac{1}{10}+\frac{1}{10}+\frac{1}{4}} = \frac{\frac{2}{10}}{\frac{9}{20}} = \frac{4}{9}$$

Let us try to achieve this probability using Bayes' rule. To apply Bayes' rule, we need to compute $\Pr[\bigcirc|\text{Bin } 1]$, $\Pr[\text{Bin } 1]$ and $\Pr[\bigcirc]$. $\Pr[\bigcirc|\text{Bin } 1]$ is the chance that we pick a white ball given that we picked bin 1, which is $\frac{2}{5}$. $\Pr[\text{ Bin } 1]$ is $\frac{1}{2}$ as given in the description of the problem. Finally, $\Pr[\bigcirc]$ can be computed using the Total Probability rule:

$$\Pr[\bigcirc] = \Pr[\bigcirc|\text{Bin } 1] \times \Pr[\text{Bin } 1] + \Pr[\bigcirc|\text{Bin } 2] \times \Pr[\text{Bin } 2] = \frac{2}{5} \times \frac{1}{2} + \frac{1}{2} \times \frac{1}{2} = \frac{9}{20}.$$

Observe that we can apply the Total Probability rule here because $\Pr[\text{ Bin } 1]$ is the complement of $\Pr[\text{ Bin } 2]$. Finally, if we plug the above values into Bayes' rule we obtain the probability that we picked bin 1 given that we picked a white ball:

$$\Pr[\text{Bin } 1|\bigcirc] = \frac{\frac{2}{5}\times\frac{1}{2}}{\frac{9}{20}} = \frac{\frac{2}{10}}{\frac{9}{20}} = \frac{4}{9}.$$

All we have done above is combined Bayes' rule and the Total Probability rule; this is also how we obtained Equation (3). We could equivalently have plugged in the appropriate values to Equation (3).

Combinations of events

In most applications of probability in Computer Science, we are interested in things like $\Pr[\bigcup_{i=1}^n A_i]$ and $\Pr[\bigcap_{i=1}^n A_i]$, where the A_i are simple events (i.e., we know, or can easily compute, the $\Pr[A_i]$). The intersection $\bigcap_i A_i$ corresponds to the logical AND of the events A_i, while the union $\bigcup_i A_i$ corresponds to their logical OR. As an example, if A_i denotes the event that a failure of type i happens in a certain system, then $\bigcup_i A_i$ is the event that the system fails.

In general, computing the probabilities of such combinations can be very difficult. In this section, we discuss some situations where it can be done. Let's start with independent events, for which intersections are quite simple to compute.

Independent Events

Definition 14.2 (Independence)**.** *Two events A, B in the same probability space are independent if* $\Pr[A \cap B] = \Pr[A] \times \Pr[B]$.

The intuition behind this definition is the following. Suppose that $\Pr[B] > 0$. Then we have

$$\Pr[A|B] = \frac{\Pr[A \cap B]}{\Pr[B]} = \frac{\Pr[A] \times \Pr[B]}{\Pr[B]} = \Pr[A].$$

Thus independence has the natural meaning that "the probability of A is not affected by whether or not B occurs." (By a symmetrical argument, we also have $\Pr[B|A] = \Pr[B]$ provided $\Pr[A] > 0$.) For events A, B such that $\Pr[B] > 0$, the condition $\Pr[A|B] = \Pr[A]$ is actually *equivalent* to the definition of independence.

Several of our previously mentioned random experiments consist of independent events. For example, if we flip a coin twice, the event of obtaining heads in the first trial is independent to the event of obtaining heads in the second trial. The same applies for two rolls of a die; the outcomes of each trial are independent.

The above definition generalizes to any finite set of events:

Definition 14.3 (Mutual independence)**.** *Events $A_1, \ldots, A_n$ are mutually independent if for every subset $I \subseteq \{1, \ldots, n\}$,*

$$\Pr[\textstyle\bigcap_{i \in I} A_i] = \prod_{i \in I} \Pr[A_i].$$

Note that we need this property to hold for *every* subset I.

For mutually independent events $A_1, \ldots, A_n$, it is not hard to check from the definition of conditional probability that, for any $1 \leq i \leq n$ and any subset $I \subseteq \{1, \ldots, n\} \setminus \{i\}$, we have

$$\Pr[A_i | \textstyle\bigcap_{j \in I} A_j] = \Pr[A_i].$$

Note that the independence of every pair of events (so-called *pairwise independence*) does *not* necessarily imply mutual independence. For example, it is possible to construct three events A, B, C such that each *pair* is independent but the triple A, B, C is *not* mutually independent.

Pairwise Independence Example

Suppose you toss a fair coin twice and let A be the event that the first flip is H's and B be the event that the second flip is H's. Now let C be the event that both flips are the same (i.e. both H's or both T's). Of course A and B are independent. What is more interesting is that so are A and C: given that the first toss came up H's, there is still an even chance that the second flip is the same as the first. Another way of saying this is that $P[A \cap C] = P[A]P[C] = 1/4$ since $A \cap C$ is the event that the first flip is H's and the second is also H's. By the same reasoning B and C are also independent. On the other hand, A, B and C are not mutually independent. For example if we are given that A and B occurred then the probability that C occurs is 1. So even though A, B and C are not mutually independent, every pair of them are independent. In other words, A, B and C are pairwise independent but not mutually independent.

Intersections of events

Computing intersections of independent events is easy; it follows from the definition. We simply multiply the probabilities of each event. How do we compute intersections for events which may not be independent? From the definition of conditional probability, we immediately have the following product rule (sometimes also called the chain rule) for computing the probability of an intersection of events.

Theorem 14.1 (Product Rule)**.** *For any events A, B, we have*

$$\Pr[A \cap B] = \Pr[A] \Pr[B|A].$$

More generally, for any events $A_1, \ldots, A_n$,

$$\Pr[\textstyle\bigcap_{i=1}^{n} A_i] = \Pr[A_1] \times \Pr[A_2|A_1] \times \Pr[A_3|A_1 \cap A_2] \times \cdots \times \Pr[A_n|\textstyle\bigcap_{i=1}^{n-1} A_i].$$

Proof. The first assertion follows directly from the definition of $\Pr[B|A]$ (and is in fact a special case of the second assertion with $n = 2$).

To prove the second assertion, we will use induction on n (the number of events). The base case is $n = 1$, and corresponds to the statement that $\Pr[A] = \Pr[A]$, which is trivially true. For the inductive step, let $n > 1$ and assume (the inductive hypothesis) that

$$\Pr[\textstyle\bigcap_{i=1}^{n-1} A_i] = \Pr[A_1] \times \Pr[A_2|A_1] \times \cdots \times \Pr[A_{n-1}|\textstyle\bigcap_{i=1}^{n-2} A_i].$$

Now we can apply the definition of conditional probability to the two events A_n and $\bigcap_{i=1}^{n-1} A_i$ to deduce that

$$\begin{aligned}\Pr[\textstyle\bigcap_{i=1}^{n} A_i] = \Pr[A_n \cap (\textstyle\bigcap_{i=1}^{n-1} A_i)] &= \Pr[A_n|\textstyle\bigcap_{i=1}^{n-1} A_i] \times \Pr[\textstyle\bigcap_{i=1}^{n-1} A_i] \\ &= \Pr[A_n|\textstyle\bigcap_{i=1}^{n-1} A_i] \times \Pr[A_1] \times \Pr[A_2|A_1] \times \cdots \times \Pr[A_{n-1}|\textstyle\bigcap_{i=1}^{n-2} A_i],\end{aligned}$$

where in the last line we have used the inductive hypothesis. This completes the proof by induction. □

The product rule is particularly useful when we can view our sample space as a sequence of choices. The next few examples illustrate this point.

Examples

Coin tosses

Toss a fair coin three times. Let A be the event that all three tosses are heads. Then $A = A_1 \cap A_2 \cap A_3$, where A_i is the event that the ith toss comes up heads. We have

$$\begin{aligned}\Pr[A] &= \Pr[A_1] \times \Pr[A_2|A_1] \times \Pr[A_3|A_1 \cap A_2] \\ &= \Pr[A_1] \times \Pr[A_2] \times \Pr[A_3] \\ &= \tfrac{1}{2} \times \tfrac{1}{2} \times \tfrac{1}{2} = \tfrac{1}{8}.\end{aligned}$$

The second line here follows from the fact that the tosses are mutually independent. Of course, we already know that $\Pr[A] = \frac{1}{8}$ from our definition of the probability space in the previous lecture note. Another way of looking at this calculation is that it justifies our definition of the probability space, and shows that it was consistent with assuming that the coin flips are mutually independent.

If the coin is biased with heads probability p, we get, again using independence,

$$\Pr[A] = \Pr[A_1] \times \Pr[A_2] \times \Pr[A_3] = p^3.$$

And more generally, the probability of any sequence of n tosses containing r heads and $n-r$ tails is $p^r(1-p)^{n-r}$. This is in fact the reason we defined the probability space this way in the previous lecture note: we defined the sample point probabilities so that the coin tosses would behave independently.

Monty Hall

Recall the Monty Hall problem from the last lecture: there are three doors and the probability that the prize is behind any given door is $\frac{1}{3}$. There are goats behind the other two doors. The contestant picks a door randomly, and the host opens one of the other two doors, revealing a goat. How do we calculate intersections in this setting? For example, what is the probability that the contestant chooses door 1, the prize is behind door 2, and the host chooses door 3?

Let A_1 be the event that the contestant chooses door 1, let A_2 be the event that the prize is behind door 2, and let A_3 be the event that the host chooses door 3. We would like to compute $\Pr[A_1 \cap A_2 \cap A_3]$. By the product rule:

$$\Pr[A_1 \cap A_2 \cap A_3] = \Pr[A_1] \times \Pr[A_2|A_1] \times \Pr[A_3|A_1 \cap A_2]$$

The probability of A_1 is $\frac{1}{3}$, since the contestant is choosing the door at random. The probability A_2 given A_1 is still $\frac{1}{3}$ since they are independent. The probability of the host choosing door 3 given events A_1 and A_2 is 1; the host cannot choose door 1, since the contestant has already opened it, and the host cannot choose door 2, since the host must reveal a goat (and not the prize). Therefore,

$$\Pr[A_1 \cap A_2 \cap A_3] = \frac{1}{3} \times \frac{1}{3} \times 1 = \frac{1}{9}.$$

Observe that we did need conditional probability in this setting; had we simply multiplied the probabilities of each event, we would have obtained $\frac{1}{27}$ since the probability of A_3 is also $\frac{1}{3}$ (can you figure out why?). What if we changed the situation, and instead asked for the probability that the contestant chooses door 1, the prize is behind door 1, and the host chooses door 2? We can use the same technique as above, but our final answer will be different. This is left as an exercise.

Poker Hands

Let's use the product rule to compute the probability of a flush in a different way. This is equal to $4 \times \Pr[A]$, where A is the probability of a Hearts flush. Intuitively, this should be clear since there are 4 suits; we'll see why this is formally true in the next section. We can write $A = \bigcap_{i=1}^{5} A_i$, where A_i is the event that the ith card we pick is a Heart. So we have

$$\Pr[A] = \Pr[A_1] \times \Pr[A_2|A_1] \times \cdots \times \Pr[A_5|\bigcap_{i=1}^{4} A_i].$$

Clearly $\Pr[A_1] = \frac{13}{52} = \frac{1}{4}$. What about $\Pr[A_2|A_1]$? Well, since we are conditioning on A_1 (the first card is a Heart), there are only 51 remaining possibilities for the second card, 12 of which are Hearts. So $\Pr[A_2|A_1] = \frac{12}{51}$. Similarly, $\Pr[A_3|A_1 \cap A_2] = \frac{11}{50}$, and so on. So we get

$$4 \times \Pr[A] = 4 \times \frac{13}{52} \times \frac{12}{51} \times \frac{11}{50} \times \frac{10}{49} \times \frac{9}{48},$$

which is exactly the same fraction we computed in the previous lecture note.

So now we have two methods of computing probabilities in many of our sample spaces. It is useful to keep these different methods around, both as a check on your answers and because in some cases one of the methods is easier to use than the other.

Unions of Events

You are in Las Vegas, and you spy a new game with the following rules. You pick a number between 1 and 6. Then three dice are thrown. You win if and only if your number comes up on at least one of the dice.

The casino claims that your odds of winning are 50%, using the following argument. Let A be the event that you win. We can write $A = A_1 \cup A_2 \cup A_3$, where A_i is the event that your number comes up on die i. Clearly $\Pr[A_i] = \frac{1}{6}$ for each i. Therefore,

$$\Pr[A] = \Pr[A_1 \cup A_2 \cup A_3] = \Pr[A_1] + \Pr[A_2] + \Pr[A_3] = 3 \times \frac{1}{6} = \frac{1}{2}.$$

Is this calculation correct? Well, suppose instead that the casino rolled six dice, and again you win iff your number comes up at least once. Then the analogous calculation would say that you win with probability $6 \times \frac{1}{6} = 1$, i.e., certainly! The situation becomes even more ridiculous when the number of dice gets bigger than 6.

The problem is that the events A_i are *not disjoint*: i.e., there are some sample points that lie in more than one of the A_i. (We could get really lucky and our number could come up on two of the dice, or all three.) So if we add up the $\Pr[A_i]$ we are counting some sample points more than once.

Fortunately, there is a formula for this, known as the Principle of Inclusion/Exclusion:

Theorem 14.2 (Inclusion/Exclusion)**.** *For events $A_1, \ldots, A_n$ in some probability space, we have*

$$\Pr[\bigcup_{i=1}^{n} A_i] = \sum_{i=1}^{n} \Pr[A_i] - \sum_{\{i,j\}} \Pr[A_i \cap A_j] + \sum_{\{i,j,k\}} \Pr[A_i \cap A_j \cap A_k] - \cdots \pm \Pr[\bigcap_{i=1}^{n} A_i].$$

[In the above summations, $\{i, j\}$ denotes all unordered pairs with $i \neq j$, $\{i, j, k\}$ denotes all unordered triples of distinct elements, and so on.]

I.e., to compute $\Pr[\bigcup_i A_i]$, we start by summing the event probabilities $\Pr[A_i]$, then we *subtract* the probabilities of all pairwise intersections, then we *add* back in the probabilities of all three-way intersections, and so on.

We won't prove this formula here; but you might like to verify it for the special case $n = 3$ by drawing a Venn diagram and checking that every sample point in $A_1 \cup A_2 \cup A_3$ is counted exactly once by the formula. You might also like to prove the formula for general n by induction (in similar fashion to the proof of the Product Rule above).

Taking the formula on faith, what is the probability we get lucky in the new game in Vegas?

$$\Pr[A_1 \cup A_2 \cup A_3] = \Pr[A_1] + \Pr[A_2] + \Pr[A_3] - \Pr[A_1 \cap A_2] - \Pr[A_1 \cap A_3] - \Pr[A_2 \cap A_3] + \Pr[A_1 \cap A_2 \cap A_3].$$

Now the nice thing here is that the events A_i are mutually independent (the outcome of any die does not depend on that of the others), so $\Pr[A_i \cap A_j] = \Pr[A_i]\Pr[A_j] = (\frac{1}{6})^2 = \frac{1}{36}$, and similarly $\Pr[A_1 \cap A_2 \cap A_3] = (\frac{1}{6})^3 = \frac{1}{216}$. So we get

$$\Pr[A_1 \cup A_2 \cup A_3] = \left(3 \times \tfrac{1}{6}\right) - \left(3 \times \tfrac{1}{36}\right) + \tfrac{1}{216} = \tfrac{91}{216} \approx 0.42.$$

So your odds are quite a bit worse than the casino is claiming!

When n is large (i.e., we are interested in the union of many events), the Inclusion/Exclusion formula is essentially useless because it involves computing the probability of the intersection of every non-empty subset of the events: and there are $2^n - 1$ of these! Sometimes we can just look at the first few terms of it and forget the rest: note that successive terms actually give us an overestimate and then an underestimate of the answer, and these estimates both get better as we go along.

However, in many situations we can get a long way by just looking at the first term:

1. **Disjoint events.** If the events A_i are all *disjoint* (i.e., no pair of them contain a common sample point — such events are also called *mutually exclusive*), then

   $$\Pr[\bigcup_{i=1}^n A_i] = \sum_{i=1}^n \Pr[A_i].$$

   [Note that we have already used this fact several times in our examples, e.g., in claiming that the probability of a flush is four times the probability of a Hearts flush — clearly flushes in different suits are disjoint events.]

2. **Union bound.** Always, it is the case that

   $$\Pr[\bigcup_{i=1}^n A_i] \leq \sum_{i=1}^n \Pr[A_i].$$

   This merely says that adding up the $\Pr[A_i]$ can only *over*estimate the probability of the union. Crude as it may seem, in the next lecture note we'll see how to use the union bound effectively in a Computer Science example.

Two Killer Applications: Hashing and Load Balancing

In this lecture we will see that the simple balls-and-bins process can be used to model a surprising range of phenomena. Recall that in this process we distribute k balls in n bins, where each ball is independently placed in a uniformly random bin. We can ask questions such as:

- How large can we choose k while ensuring that with probability at least $1/2$, no two balls land in the same bin?
- If $k = n$, what is the maximum number of balls that are likely to land in the same bin?

As we shall see, these two simple questions provide important insights into two important computer science applications: hashing and load balancing. Our answers to these two questions are based on the application of a simple technique called the union bound, which states that $P[A \cup B] \leq P[A] + P[B]$.

One of the basic issues in hashing is the tradeoff between the size of the hash table and the number of collisions. Here is a simple question that quantifies the tradeoff:

- Suppose a hash function distributes keys evenly over a table of size n. How many keys can we hash before the probability of a collision exceeds (say) $\frac{1}{2}$?

Recall that a hash table is a data structure that supports the storage of a set of keys drawn from a large universe U (say, the names of all 250m people in the US). The set of keys to be stored changes over time, and so the data structure allows keys to be added and deleted quickly. It also rapidly answers given a key whether it is an element in the currently stored set. The crucial question is how large must the hash table be to allow these operations (addition, deletion and membership) to be implemented quickly.

Here is how the hashing works. The hash function h maps U to a table T of modest size. To ADD a key x to our set, we evaluate $h(x)$ (i.e., apply the hash function to the key) and store x at the location $h(x)$ in the table T. All keys in our set that are mapped to the same table location are stored in a simple linked list. The operations DELETE and MEMBER are implemented in similar fashion, by evaluating $h(x)$ and searching the linked list at $h(x)$. Note that the efficiency of a hash function depends on having only few collisions — i.e., keys that map to the same location. This is because the search time for DELETE and MEMBER operations is proportional to the length of the corresponding linked list.

Of course, we could be unlucky and choose keys such that our hash function maps many of them to the same location in the table. But the whole idea behind hashing is that we select our hash function carefully, so that it scrambles up the input key and seems to map it to a random location in the table, making it unlikely that most of the keys we select are mapped to the same location. To quantitatively understand this phenomenon, we will model our hash function as a random function - one that maps each key to a uniformly random location in the table, independently of where all other keys are mapped. The question we will answer is the following: what is the largest number, m, of keys we can store before the probability of a collision reaches $\frac{1}{2}$?

Note that there is nothing special about $\frac{1}{2}$. One can ask, and answer, the same question with different values of collision probability, and the largest number of keys m will change accordingly.

Balls and Bins

Let's begin by seeing how this problem can be put into the balls and bins framework. The balls will be the m keys to be stored, and the bins will be the n locations in the hash table T. Since the hash function maps each key to a random location in the table T, we can see each key (ball) as choosing a hash table location (bin) uniformly and independently from T. Thus the probability space corresponding to this hashing experiment is exactly the same as the balls and bins space.

We are interested in the event A that there is no collision, or equivalently, that all m balls land in different bins. Clearly $\Pr[A]$ will decrease as m increases (with n fixed). Our goal is to find the largest value of m such that $\Pr[A]$ remains above $\frac{1}{2}$. [Note: Really we are looking at different sample spaces here, one for each value of m. So it would be more correct to write $\Pr_m$ rather than just Pr, to make clear which sample space we are talking about. However, we will omit this detail.]

Using the union bound

Let us see how to use the union bound to achieve this goal. We will fix the value of m and try to compute $\Pr[A]$. There are exactly $k = \binom{m}{2} = \frac{m(m-1)}{2}$ possible pairs among our m keys. Imagine these are numbered from 1 to $\binom{m}{2}$ (it doesn't matter how). Let A_i denote the event that pair i has a collision (i.e., both keys in the pair are hashed to the same location). Then the event $\overline{A}$ that *some* collision occurs can be written $\overline{A} = \bigcup_{i=1}^{k} A_i$. What is $\Pr[A_i]$? We claim it is just $\frac{1}{n}$ for every i; this is just the probability that two particular balls land in the same bin.

So, using the union bound from the last lecture, we have

$$\Pr[\overline{A}] \leq \sum_{i=1}^{k} \Pr[A_i] = k \times \frac{1}{n} = \frac{m(m-1)}{2n} \approx \frac{m^2}{2n}.$$

This means that the probability of having a collision is less than $\frac{1}{2}$ provided $\frac{m^2}{2n} \leq \frac{1}{2}$, i.e., provided $m \leq \sqrt{n}$. Thus, if we wish to suffer no collisions, the size of the hash table must be about the square of the cardinality of the set we are trying to store. We will see in the next section on load balancing that the number of collisions does not increase dramatically as we decrease the size of the hash table and make it comparable to the size of the set we are trying to store.

It turns out we can derive a slightly less restrictive bound for m using other techniques from the past several lectures. Although this alternate bound is a little better, both bounds are the same in terms of dependence on n (both are of the form $m = O(\sqrt{n})$). If you are interested in the alternate derivation, please read the section at the end of this note.

Birthday Paradox

Can we do better than the $m = O(\sqrt{n})$ dependence on n? It turns out that the answer is no, and this is related to a surprising phenomenon called the birthday paradox. Suppose there are 23 students in class. What is the chance that two of them have the same birthday? Naively one might answer that since there are 365 days in the year, the chance should be roughly 23/365 $\tilde{6}$%. The correct answer is over 50%!

Suppose there are k people in the room, and let $A =$ "At least two people have the same birthday," and let $\bar{A} =$ "No two people have the same birthday." It turns out it is easier to calculate $\Pr[\bar{A}]$, and then $\Pr[A] = 1 - \Pr[\bar{A}]$.

Our sample space is of cardinality $|\Omega| = 365^k$. And the number of sample points such that no two people to have the same birthday can be calculated as follows: there are 365 choices for the first person, 364 for the second, ..., $365-k+1$ choices for the k^{th} person, for a total of $365 \times 364 \times \cdots \times (365-k+1)$. Thus $\Pr[\bar{A}] = \frac{|\bar{A}|}{|\Omega|} = \frac{365\times 364\times\cdots\times(365-k+1)}{365^k}$. And $\Pr[A] = 1 - \frac{365\times 364\times\cdots\times(365-k+1)}{365^k}$. Substituting $k = 23$, we can check that $\Pr[A]$ is over 50%. And with $k = 60$ $\Pr[A]$ is larger than 99%!

The hashing problem we considered above is a closely related to the birthday problem. Indeed, the birthday problem is the special case of the chasing problem with 365 bins. The $k = 23$ solution to the birthday problem can be seen as k being roughly $\sqrt{365}$.

Application 2: Load balancing

An important practical issue in distributed computing is how to spread the workload in a distributed system among its processors. Here we investigate an extremely simple scenario that is both fundamental in its own right and also establishes a baseline against which more sophisticated methods should be judged.

Suppose we have m identical jobs and n identical processors. Our task is to assign the jobs to the processors in such a way that no processor is too heavily loaded. Of course, there is a simple optimal solution here: just divide up the jobs as evenly as possible, so that each processor receives either $\lceil \frac{m}{n} \rceil$ or $\lfloor \frac{m}{n} \rfloor$ jobs. However, this solution requires a lot of centralized control, and/or a lot of communication: the workload has to be balanced evenly either by a powerful centralized scheduler that talks to all the processors, or by the exchange of many messages between jobs and processors. This kind of operation is very costly in most distributed systems. The question therefore is: What can we do with little or no overhead in scheduling and communication cost?

Back to Balls and Bins

The first idea that comes to mind here is... balls and bins! In other words, each job simply selects a processor uniformly at random and independently of all others, and goes to that processor. (Make sure you believe that the probability space for this experiment is the same as the one for balls and bins.) This scheme requires no communication. However, presumably it won't in general achieve an optimal balancing of the load. Let A_k be the event that the load of some processor is at least k. As designers or users of this load balancing scheme, here's one question we might care about:

Question: Find the smallest value k such that $\Pr[A_k] \leq \frac{1}{2}$.

If we have such a value k, then we'll know that, with good probability (at least $\frac{1}{2}$), every processor in our system will have a load at most k. This will give us a good idea about the performance of the system. Of course, as with our hashing application, there's nothing special about the value $\frac{1}{2}$; we're just using this for illustration. As you can check later (if you choose to read the optional section below), essentially the same analysis can be used to find k such that $\Pr[A_k] \leq 0.05$ (i.e., 95% confidence), or any other value we like. Indeed, we can even find the k's for several different confidence levels and thus build up a more detailed picture of the behavior of the scheme. To simplify our problem, we'll also assume from now on that $m = n$ (i.e., the number of jobs is the same as the number of processors). With a bit more work, we could generalize our analysis to other values of m.

Applying the Union Bound

From Application 1 we know that we get collisions already when $m \approx 1.177\sqrt{n}$. So when $m = n$ the maximum load will certainly be larger than 1 (with good probability). But how large will it be? If we try to

analyze the maximum load directly, we run into the problem that it depends on the number of jobs at *every* processor (or equivalently, the number of balls in every bin). Since the load in one bin depends on those in the others, this becomes very tricky. Instead, what we'll do is analyze the load in any *one* bin, say bin 1; this will be fairly easy. Let $A_k(1)$ be the event that the load in bin 1 is at least k. What we'll do is find k such that

$$\Pr[A_k(1)] \leq \tfrac{1}{2n}. \tag{1}$$

Since all the bins are identical, we will then know that, for the same k,

$$\Pr[A_k(i)] \leq \tfrac{1}{2n} \qquad \text{for } i = 1,2,\ldots,n,$$

where $A_k(i)$ is the event that the load in bin i is at least k. But now, since the event A_k is exactly the union of the events $A_k(i)$ (do you see why?), we can use the "Union Bound" from the previous lecture:

$$\begin{aligned}\Pr[A_k] &= \Pr[\textstyle\bigcup_{i=1}^{n} A_k(i)] \\ &\leq n \times \frac{1}{2n} \\ &= \frac{1}{2}.\end{aligned}$$

It's worth standing back to notice what we did here: we wanted to conclude that $\Pr[A_k] \leq \frac{1}{2}$. We couldn't analyze A_k directly, but we knew that $A_k = \bigcup_{i=1}^{n} A_k(i)$, for much simpler events $A_k(i)$. Since there are n events $A_k(i)$, and all have the same probability, it is enough for us to show that $\Pr[A_k(i)] \leq \frac{1}{2n}$; the union bound then guarantees that $\Pr[A_k] \leq \frac{1}{2}$. This kind of reasoning is very common in applications of probability in Computer Science.

Now all that's left to do is find k which satisfies equation 1. That is, we wish to bound the probability that bin 1 has at least k balls (and find a value of k so that this probability is smaller than $1/2n$). We start by observing that for the event $A_k(1)$ to occur (that bin 1 has at least k balls), there must be some subset S of exactly k balls such that all balls in S ended up in bin 1. We can say this more formally as follows: for a subset S (where $|S| = k$), let B_S be the event that all balls in S land in bin 1. Then the event $A_k(1)$ is a subset of the event $\cup_S B_S$ (where the union is over all sets S of cardinality k). It follows that:

$$\Pr[A_k(1)] \leq \Pr[\cup_S B_S]$$

We can use the union bound on $\Pr[\cup_S B_S]$:

$$\Pr[\cup_S B_S] \leq \sum_S \Pr[B_S]$$

There are $\binom{n}{k}$ sets we are summing over, and for each set S, $\Pr[B_S]$ is simple: it is just the probability that k balls land in bin 1, or $\frac{1}{n^k}$. Using these observations and the above equations, we can compute an upper bound on $\Pr[A_k(1)]$:

$$\Pr[A_k(1)] \leq \binom{n}{k}\frac{1}{n^k}$$

Now to satisfy our original goal (equation 1), we just need to choose k so that $\binom{n}{k}\frac{1}{n^k} \leq \frac{1}{2n}$. But we have

$$\binom{n}{k}\frac{1}{n^k} = \frac{n(n-1)\cdots(n-k+1)}{k!}\frac{1}{n^k} \leq \frac{1}{k!}$$

Setting $k! = 2n$, and simplifying we get that $\left(\frac{k}{e}\right)^k = 2n$. Taking logs we get that $k(\ln k - 1) = \ln 2n$. This gives a value of k of roughly $\frac{\ln n}{\ln \ln n}$.

If you would like to learn more about how to do this, please refer to the additional section on load balancing below.

Finally, here is one punchline from Application 2. Let's say the total US population is about 250 million. Suppose we mail 250 million items of junk mail, each one with a random US address. Then (see the optional section below for more details) with probability at least $\frac{1}{2}$, no one person anywhere will receive more than about a dozen items!

More About Hashing (Optional)

Please read on only if interested. In this section, we will derive the alternate bound described in the hashing section above.

Main Idea

Let's fix the value of m and try to compute $\Pr[A]$. Since our probability space is uniform (each outcome has probability $\frac{1}{n^m}$), it's enough just to count the number of outcomes in A. In how many ways can we arrange m balls in n bins so that no bin contains more than one ball? Well, this is just the number of ways of choosing m things out of n *without* replacement, which as we saw in Note 10 is

$$n \times (n-1) \times (n-2) \times \cdots \times (n-m+2) \times (n-m+1).$$

This formula is valid as long as $m \leq n$: if $m > n$ then clearly the answer is zero. From now on, we'll assume that $m \leq n$.

Now we can calculate the probability of no collision:

$$\begin{aligned} \Pr[A] &= \frac{n(n-1)(n-2)\dots(n-m+1)}{n^m} \\ &= \frac{n}{n} \times \frac{n-1}{n} \times \frac{n-2}{n} \times \cdots \times \frac{n-m+1}{n} \\ &= \left(1-\frac{1}{n}\right) \times \left(1-\frac{2}{n}\right) \times \cdots \times \left(1-\frac{m-1}{n}\right). \end{aligned} \tag{2}$$

Before going on, let's pause to observe that we could compute $\Pr[A]$ in a different way, as follows. View the probability space as a sequence of choices, one for each ball. For $1 \leq i \leq m$, let A_i be the event that the ith ball lands in a different bin from balls $1, 2, \dots, i-1$. Then

$$\begin{aligned} \Pr[A] = \Pr[\textstyle\bigcap_{i=1}^{n} A_i] &= \Pr[A_1] \times \Pr[A_2|A_1] \times \Pr[A_3|A_1 \cap A_2] \times \cdots \times \Pr[A_m|\textstyle\bigcap_{i=1}^{m-1} A_i] \\ &= 1 \times \frac{n-1}{n} \times \frac{n-2}{n} \times \cdots \times \frac{n-m+1}{n} \\ &= \left(1-\frac{1}{n}\right) \times \left(1-\frac{2}{n}\right) \times \cdots \times \left(1-\frac{m-1}{n}\right). \end{aligned}$$

Fortunately, we get the same answer as before! [You should make sure you see how we obtained the conditional probabilities in the second line above. For example, $\Pr[A_3|A_1 \cap A_2]$ is the probability that the third ball lands in a different bin from the first two balls, *given that* those two balls also landed in different bins. This means that the third ball has $n-2$ possible bin choices out of a total of n.]

Essentially, we are now done with our problem: equation (2) gives an exact formula for the probability of no collision when m keys are hashed. All we need to do now is plug values $m = 1, 2, 3, \dots$ into (2) until we find that $\Pr[A]$ drops below $\frac{1}{2}$. The corresponding value of m (minus 1) is what we want.

We can actually make this bound much more useful by turning it around, as we will do below. We will derive an equation which tells us the value of m at which $\Pr[A]$ drops below $\frac{1}{2}$. It turns out that if m is smaller than approximately $1.177\sqrt{n}$, the probability of a collision will be less than $\frac{1}{2}$.

Further Simplification

The bound we gave above (for the largest number m of keys we can store before the probability of a collision reaches $\frac{1}{2}$) is not really satisfactory: it would be much more useful to have a formula that gives the "critical" value of m directly, rather than having to compute $\Pr[A]$ for $m = 1, 2, 3, \ldots$. Note that we would have to do this computation separately for each different value of n we are interested in: i.e., whenever we change the size of our hash table.

So what remains is to "turn equation (2) around", so that it tells us the value of m at which $\Pr[A]$ drops below $\frac{1}{2}$. To do this, let's take logs: this is a good thing to do because it turns the product into a sum, which is easier to handle. We get

$$\ln(\Pr[A]) = \ln\Big(1 - \frac{1}{n}\Big) + \ln\Big(1 - \frac{2}{n}\Big) + \cdots + \ln\Big(1 - \frac{m-1}{n}\Big), \tag{3}$$

where "ln" denotes natural (base e) logarithm. Now we can make use of a standard approximation for logarithms: namely, if x is small then $\ln(1-x) \approx -x$. This comes from the Taylor series expansion

$$\ln(1-x) = -x - \frac{x^2}{2} - \frac{x^3}{3} - \ldots.$$

So by replacing $\ln(1-x)$ by $-x$ we are making an error of at most $(\frac{x^2}{2} + \frac{x^3}{3} + \cdots)$, which is at most $2x^2$ when $x \leq \frac{1}{2}$. In other words, we have

$$-x \geq \ln(1-x) \geq -x - 2x^2.$$

And if x is small then the error term $2x^2$ will be much smaller than the main term $-x$. Rather than carry around the error term $2x^2$ everywhere, in what follows we'll just write $\ln(1-x) \approx -x$, secure in the knowledge that we could make this approximation precise if necessary.

Now let's plug this approximation into equation (3):

$$\begin{aligned}
\ln(\Pr[A]) &\approx -\frac{1}{n} - \frac{2}{n} - \frac{3}{n} - \ldots - \frac{m-1}{n} \\
&= -\frac{1}{n}\sum_{i=1}^{m-1} i \\
&= -\frac{m(m-1)}{2n} \\
&\approx -\frac{m^2}{2n}.
\end{aligned} \tag{4}$$

Note that we've used the approximation for $\ln(1-x)$ with $x = \frac{1}{n}, \frac{2}{n}, \frac{3}{n}, \ldots, \frac{m-1}{n}$. So our approximation should be good provided all these are small, i.e., provided n is fairly big and m is quite a bit smaller than n. Once we're done, we'll see that the approximation is actually pretty good even for modest sizes of n.

Now we can undo the logs in (4) to get our expression for $\Pr[A]$:

$$\Pr[A] \approx \mathrm{e}^{-\frac{m^2}{2n}}.$$

The final step is to figure out for what value of m this probability becomes $\frac{1}{2}$. So we want the largest m such that $e^{-\frac{m^2}{2n}} \geq \frac{1}{2}$. This means we must have

$$-\frac{m^2}{2n} \geq \ln(\tfrac{1}{2}) = -\ln 2, \tag{5}$$

or equivalently

$$m \leq \sqrt{(2\ln 2)n} \approx 1.177\sqrt{n}. \tag{6}$$

So the bottom line is that we can hash approximately $\underline{m = \lfloor 1.177\sqrt{n} \rfloor \text{ keys}}$ before the probability of a collision reaches $\frac{1}{2}$.

Recall that our calculation was only approximate; so we should go back and get a feel for how much error we made. We can do this by using equation (2) to compute the exact value $m = m_0$ at which $\Pr[A]$ drops below $\frac{1}{2}$, for a few sample values of n. Then we can compare these values with our estimate $m = 1.177\sqrt{n}$.

n	10	20	50	100	200	365	500	1000	10^4	10^5	10^6
$1.177\sqrt{n}$	3.7	5.3	8.3	11.8	16.6	22.5	26.3	37.3	118	372	1177
exact m_0	4	5	8	12	16	22	26	37	118	372	1177

From the table, we see that our approximation is very good even for small values of n. When n is large, the error in the approximation becomes negligible.

Why $\frac{1}{2}$?

As mentioned above, we could consider values other than $\frac{1}{2}$. What we did above was to (approximately) compute $\Pr[A]$ (the probability of no collision) as a function of m, and then find the largest value of m for which our estimate is smaller than $\frac{1}{2}$. If instead we were interested in keeping the collision probability below (say) 0.05 (= 5%), we could derive that we could hash at most $m = \sqrt{(2\ln(20/19))n} \approx 0.32\sqrt{n}$ keys. Of course, this number is a bit smaller than before because our collision probability is now smaller. But no matter what "confidence" probability we specify, our critical value of m will always be $c\sqrt{n}$ for some constant c (which depends on the confidence).

More About Load Balancing (Optional)

In this section, we'll come up with a slightly tighter bound for k and we will also show how to choose k so that the probability of an overloaded processor is less than $\frac{1}{2}$.

We'll start with an alternate calculation of $\Pr[A_k(1)]$. Let $C_j(1)$ be the event that the number of balls in bin 1 is exactly j. It's not hard to write down an *exact* expression for $\Pr[C_j(1)]$]:

$$\Pr[C_j(1)] = \binom{n}{j} \left(\tfrac{1}{n}\right)^j \left(1 - \tfrac{1}{n}\right)^{n-j}. \tag{7}$$

This can be seen by viewing each ball as a biased coin toss: Heads corresponds to the ball landing in bin 1, Tails to all other outcomes. So the Heads probability is $\frac{1}{n}$; and all coin tosses are (mutually) independent. As we saw in earlier lectures, (7) gives the probability of exactly j Heads in n tosses.

Thus we have

$$\Pr[A_k(1)] = \sum_{j=k}^{n} \Pr[C_j(1)] = \sum_{j=k}^{n} \binom{n}{j} \left(\tfrac{1}{n}\right)^j \left(1-\tfrac{1}{n}\right)^{n-j}. \tag{8}$$

Now in some sense we are done: we could try plugging values $k = 1, 2, \ldots$ into (8) until the probability drops below $\frac{1}{2n}$. However, it is also possible to massage equation (8) into a cleaner form from which we can read off the value of k more directly. We will need to do a few calculations and approximations:

$$\begin{aligned}
\Pr[A_k(1)] &= \sum_{j=k}^{n} \binom{n}{j} \left(\tfrac{1}{n}\right)^j \left(1-\tfrac{1}{n}\right)^{n-j} \\
&\leq \sum_{j=k}^{n} \left(\tfrac{n\mathrm{e}}{j}\right)^j \left(\tfrac{1}{n}\right)^j \\
&= \sum_{j=k}^{n} \left(\tfrac{\mathrm{e}}{j}\right)^j
\end{aligned} \tag{9}$$

In the second line here, we used the standard approximation[1] $(\frac{n}{j})^j \leq \binom{n}{j} \leq (\frac{n\mathrm{e}}{j})^j$. Also, we tossed away the $(1-\frac{1}{n})^{n-j}$ term, which is permissible because doing so can only increase the value of the sum (i.e., since $(1-\frac{1}{n})^{n-j} \leq 1$). It will turn out that we didn't lose too much by applying these bounds.

Now we're already down to a much cleaner form in (9). To finish up, we just need to sum the series. Again, we'll make an approximation to simplify our task, and hope that it doesn't cost us too much. The terms in the series in (9) go down at each step by a factor of at least $\frac{\mathrm{e}}{k}$. So we can bound the series by a *geometric* series, which is easy to sum:

$$\sum_{j=k}^{n} \left(\tfrac{\mathrm{e}}{j}\right)^j \leq \left(\tfrac{\mathrm{e}}{k}\right)^k \left(1 + \tfrac{\mathrm{e}}{k} + \left(\tfrac{\mathrm{e}}{k}\right)^2 + \cdots\right) \leq 2\left(\tfrac{\mathrm{e}}{k}\right)^k, \tag{10}$$

where the second inequality holds provided we assume that $k \geq 2\mathrm{e}$ (which it will be, as we shall see in a moment).

So what we have now shown is the following:

$$\Pr[A_k(1)] \leq 2\left(\tfrac{\mathrm{e}}{k}\right)^k \tag{11}$$

(provided $k \geq 2\mathrm{e}$). Note that, even though we have made a few approximations, inequality (11) is completely valid: all our approximations were "$\leq$", so we always have an *upper* bound on $\Pr[X_1 \geq k]$. [You should go back through all the steps and check this.]

Recall from (1) that our goal is to make the probability in (11) less than $\frac{1}{2n}$. We can ensure this by choosing k so that

$$\left(\tfrac{\mathrm{e}}{k}\right)^k \leq \tfrac{1}{4n}. \tag{12}$$

Now we are in good shape: given any value n for the number of jobs/processors, we just need to find the smallest value $k = k_0$ that satisfies inequality (12). We will then know that, with probability at least $\frac{1}{2}$, the maximum load on any processor is at most k_0. The table below shows the values of k_0 for some sample values of n. It is recommended that you perform the experiment and compare these values of k_0 with what happens in practice.

[1] Computer scientists and mathematicians carry around a little bag of tricks for replacing complicated expressions like $\binom{n}{j}$ with simpler approximations. This is just one of these. It isn't too hard to prove the lower bound, i.e., that $\binom{n}{j} \geq (\frac{n}{j})^j$. The upper bound is a bit trickier, and makes use of another approximation for $n!$ known as *Stirling's approximation*, which implies that $j! \geq (\frac{j}{\mathrm{e}})^j$. We won't discuss the details here.

n	10	20	50	100	500	1000	10^4	10^5	10^6	10^7	10^8	10^{15}
exact k_0	6	6	7	7	8	8	9	10	11	12	13	19
$\ln(4n)$	3.7	4.4	5.3	6.0	7.6	8.3	10.6	13.9	15.2	17.5	19.8	36
$\frac{2\ln n}{\ln\ln n}$	5.6	5.4	5.8	6.0	6.8	7.2	8.2	9.4	10.6	11.6	12.6	20

Can we come up with a formula for k_0 as a function of n (as we did for the hashing problem)? Well, let's take logs in (12):

$$k(\ln k - 1) \geq \ln(4n). \tag{13}$$

From this, we might guess that $k = \ln(4n)$ is a good value for k_0. Plugging in this value of k makes the left-hand side of (13) equal to $\ln(4n)(\ln\ln(4n) - 1)$, which is certainly bigger than $\ln(4n)$ provided $\ln\ln(4n) \geq 2$, i.e., $n \geq \frac{1}{4}e^{e^2} \approx 405$. So for $n \geq 405$ we can claim that the maximum load is (with probability at least $\frac{1}{2}$) no larger than $\ln(4n)$. The table above plots the values of $\ln(4n)$ for comparison with k_0. As expected, the estimate is quite good for small n, but becomes rather pessimistic when n is large.

For large n we can do better as follows. If we plug the value $k = \frac{\ln n}{\ln\ln n}$ into the left-hand side of (13), it becomes

$$\frac{\ln n}{\ln\ln n}(\ln\ln n - \ln\ln\ln n - 1) = \ln n\left(1 - \frac{\ln\ln\ln n + 1}{\ln\ln n}\right). \tag{14}$$

Now when n is large this is *just barely* smaller than the right-hand side, $\ln(4n)$. Why? Because the second term inside the parentheses goes to zero as $n \to \infty$,[2] and because $\ln(4n) = \ln n + \ln 4$, which is very close to $\ln n$ when n is large (since $\ln 4$ is a fixed small constant). So we can conclude that, for large values of n, the quantity $\frac{\ln n}{\ln\ln n}$ should be a pretty good estimate of k_0. Actually for this estimate to become good n has to be (literally) astronomically large. For more civilized values of n, we get a better estimate by taking $k = \frac{2\ln n}{\ln\ln n}$. The extra factor of 2 helps to wipe out the lower order terms (i.e., the second term in the parenthesis in (14) and the $\ln 4$) more quickly. The table above also shows the behavior of this estimate for various values of n.

[2]To see this, note that it is of the form $\frac{\ln z+1}{z}$ where $z = \ln\ln n$, and of course $\frac{\ln z+1}{z} \to 0$ as $z \to \infty$.

Random Variables: Distribution and Expectation

Example: Coin Flips

Recall our setup of a probabilistic experiment as a procedure of drawing a sample from a set of possible values, and assigning a probability for each possible outcome of the experiment. For example, if we toss a fair coin n times, then there are 2^n possible outcomes, each of which is equally likely and has probability $\frac{1}{2^n}$.

Now suppose we want to make a measurement in our experiment. For example, we can ask what is the number of heads in n coin tosses; call this number X. Of course, X is not a fixed number, but it depends on the actual sequence of coin flips that we obtain. For example, if $n = 4$ and we observe the outcome $\omega = HTHH$, then the number of heads is $X = 3$; whereas if we observe the outcome $\omega = HTHT$, then the number of heads is $X = 2$. In this example of n coin tosses we only know that X is an integer between 0 and n, but we do not know what its exact value is until we observe which outcome of n coin flips is realized and count how many heads there are. Because every possible outcome is assigned a probability, the value X also carries with it a probability for each possible value it can take. The table below lists all the possible values X can take in the example of $n = 4$ coin tosses, along with their respective probabilities.

outcomes ω	value of X (number of heads)	probability occurring
TTTT	0	1/16
HTTT, THTT, TTHT, TTTH	1	4/16
HHTT, HTHT, HTTH, THHT, THTH, TTHH	2	6/16
HHHT, HHTH, HTHH, THHH	3	4/16
HHHH	4	1/16

Such a value X that depends on the outcome of the probabilistic experiment is called a *random variable* (or *r.v.*). As we see from the example above, X does not have a definitive value, but instead only has a probability *distribution* over the set of possible values X can take, which is why it is called random. So the question "What is the number of heads in n coin tosses?" does not exactly make sense because the answer X is a random variable. But the question "What is the *typical* number of heads in n coin tosses?" makes sense: it is asking what is the average value of X (the number of heads) if we repeat the experiment of tossing n coins multiple times. This average value is called the *expectation* of X, and is one of the most useful summary (also called *statistics*) of an experiment.

Example: Permutations

Before we formalize all these notions, let us consider another example to enforce our conceptual understanding of a random variable. Suppose we collect the homeworks of n students, randomly shuffle them, and return them to the students. How many students receive their own homework?

Here the probability space consists of all $n!$ permutations of the homeworks, each with equal probability $\frac{1}{n!}$. If we label the homeworks as $1, 2, \ldots, n$, then each sample point is a permutation $\pi = (\pi_1, \ldots, \pi_n)$ where π_i

is the homework that is returned to the i-th student. Note that $\pi_1, \ldots, \pi_n \in \{1, 2, \ldots, n\}$ are all distinct, so each element in $\{1, \ldots, n\}$ appears exactly once in the permutation π.

In this setting, the i-th student receives her own homework if and only if $\pi_i = i$. Then the question "How many students receive their own homework?" translates into the question of how many indices i's satisfy $\pi_i = i$. These are known as fixed points of the permutation. As in the coin flipping case above, our question does not have a simple numerical answer (such as 4), because the number depends on the particular permutation we choose (i.e., on the sample point). Let's call the number of fixed points X, which is a random variable.

To illustrate the idea concretely, let's consider the example $n = 3$. The following table gives a complete listing of the sample space (of size $3! = 6$), together with the corresponding value of X for each sample point. Here we see that X takes on values 0, 1 or 3, depending on the sample point. You should check that you agree with this table.

permutation π	value of X (number of fixed points)
123	3
132	1
213	1
231	0
312	0
321	1

Random Variables

Let us formalize the concepts discussed above.

Definition 16.1 (Random Variable)**.** *A random variable X on a sample space Ω is a function $X : \Omega \to \mathbb{R}$ that assigns to each sample point $\omega \in \Omega$ a real number $X(\omega)$.*

Until further notice, we will restrict out attention to random variables that are discrete, i.e., they take values in a range that is finite or countably infinite. This means even though we define X to map Ω to $\mathbb{R}$, the actual set of values $\{X(\omega) : \omega \in \Omega\}$ that X takes is a discrete subset of $\mathbb{R}$.

A random variable can be visualized in general by the picture in Figure 1.[1] Note that the term "random variable" is really something of a misnomer: it is a function so there is nothing random about it and it is definitely not a variable! What is random is which sample point of the experiment is realized and hence the value that the random variable maps the sample point to.

Distribution

When we introduced the basic probability space in Note 13, we defined two things:

1. the sample space Ω consisting of all the possible outcomes (sample points) of the experiment;
2. the probability of each of the sample points.

[1] The figures in this note are inspired by figures in Chapter 2 of "Introduction to Probability" by D. Bertsekas and J. Tsitsiklis.

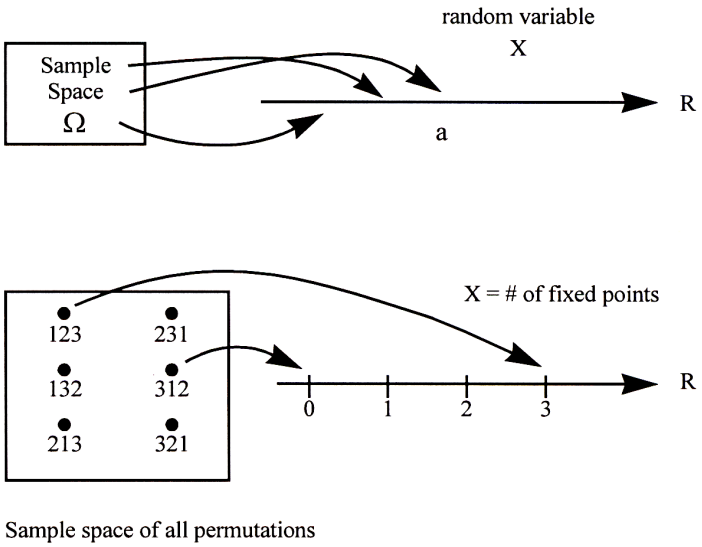


Figure 1: Visualization of how a random variable is defined on the sample space.

Analogously, there are two things important about any random variable:

1. the set of values that it can take;
2. the probabilities with which it takes on the values.

Since a random variable is defined on a probability space, we can calculate these probabilities given the probabilities of the sample points. Let a be any number in the range of a random variable X. Then the set

$$\{\omega \in \Omega : X(\omega) = a\}$$

is an *event* in the sample space (simply because it is a subset of Ω). We usually abbreviate this event to simply "$X = a$". Since $X = a$ is an event, we can talk about its probability, $\Pr[X = a]$. The collection of these probabilities, for all possible values of a, is known as the *distribution* of the random variable X.

Definition 16.2 (Distribution)**.** *The distribution of a discrete random variable X is the collection of values $\{(a, \Pr[X = a]) : a \in \mathscr{A}\}$, where $\mathscr{A}$ is the set of all possible values taken by X.*

Thus, the distribution of the random variable X in our permutation example above is:

$$\Pr[X = 0] = \frac{1}{3}; \qquad \Pr[X = 1] = \frac{1}{2}; \qquad \Pr[X = 3] = \frac{1}{6};$$

and $\Pr[X = a] = 0$ for all other values of a.

The distribution of a random variable can be visualized as a bar diagram, shown in Figure 2. The x-axis represents the values that the random variable can take on. The height of the bar at a value a is the probability $\Pr[X = a]$. Each of these probabilities can be computed by looking at the probability of the corresponding event in the sample space.

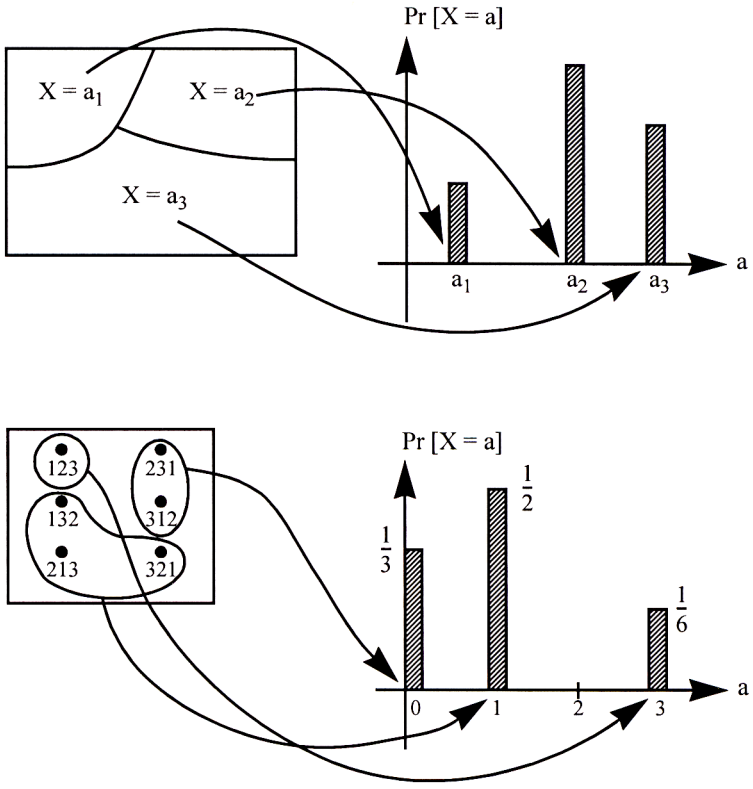


Figure 2: Visualization of how the distribution of a random variable is defined.

Note that the collection of events $X = a$, $a \in \mathscr{A}$, satisfy two important properties:

- Any two events $X = a_1$ and $X = a_2$ with $a_1 \neq a_2$ are disjoint.
- The union of all these events is equal to the entire sample space Ω.

The collection of events thus form a *partition* of the sample space (see Figure 2). Both properties follow directly from the fact that X is a function defined on Ω, i.e., X assigns a unique value to each and every possible sample point in Ω. As a consequence, the sum of the probabilities $\Pr[X = a]$ over all possible values of a is exactly 1. So when we sum up the probabilities of the events $X = a$, we are really summing up the probabilities of all the sample points.

Example: The Binomial Distribution

Let's return to our coin tossing example above, where we defined our random variable X to be the number of heads. More formally, consider the random experiment consisting of n independent tosses of a biased coin which lands on heads with probability p. Each sample point ω is a sequence of tosses. $X(\omega)$ is defined to be the number of heads in ω. For example, when $n = 3$, $X(THH) = 2$.

To compute the distribution of X, we first enumerate the possible values X can take on. They are simply $0, 1, \ldots, n$. Then we compute the probability of each event $X = i$ for $i = 0, 1, \ldots, n$. The probability of the event $X = i$ is the sum of the probabilities of all the sample points with exactly i heads (for example, if $n = 3$ and $i = 2$, there would be three such sample points $\{HHT, HTH, THH\}$). Any such sample point has a

probability $p^i(1-p)^{n-i}$, since the coin flips are independent. There are exactly $\binom{n}{i}$ of these sample points. So

$$\Pr[X=i] = \binom{n}{i} p^i (1-p)^{n-i} \qquad \text{for } i = 0, 1, \ldots, n. \tag{1}$$

This distribution, called the *binomial* distribution, is one of the most important distributions in probability. A random variable with this distribution is called a *binomial* random variable, written as

$$X \sim Bin(n, p)$$

An example of a binomial distribution is shown in Figure 3. Notice that due to the properties of X mentioned above, it must be the case that $\sum_{i=0}^{n} \Pr[X=i] = 1$, which implies that $\sum_{i=0}^{n} \binom{n}{i} p^i (1-p)^{n-i} = 1$. This provides a probabilistic proof of the Binomial Theorem!

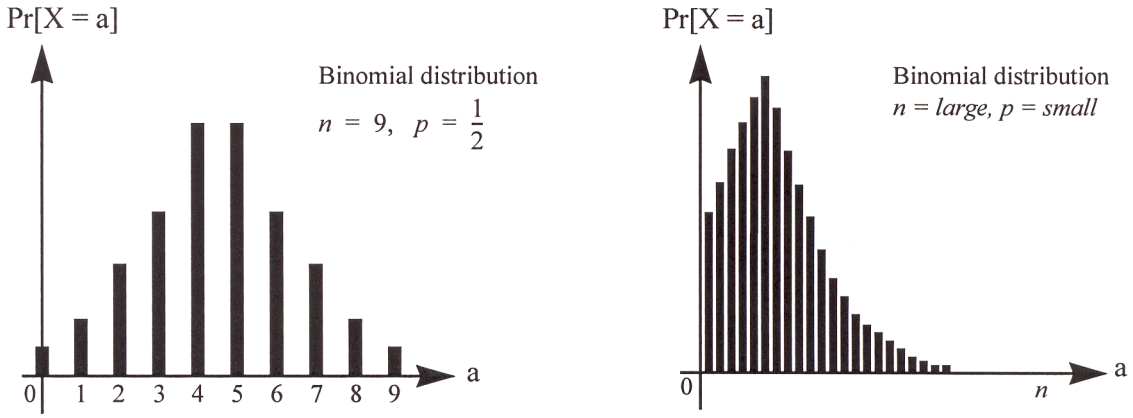


Figure 3: The binomial distributions for two choices of (n, p).

Although we define the binomial distribution in terms of an experiment involving tossing coins, this distribution is useful for modeling many real-world problems. Consider for example the error correction problem studied in Note 9. Recall that we wanted to encode n packets into $n+k$ packets such that the recipient can reconstruct the original n packets from any n packets received. But in practice, the number of packet losses is random, so how do we choose k, the amount of redundancy? If we model each packet getting lost with probability p and the losses are independent, then if we transmit $n+k$ packets, the number of packets received is a random variable X with binomial distribution: $X \sim Bin(n+k, 1-p)$ (we are tossing a coin $n+k$ times, and each coin turns out to be a head (packet received) with probability $1-p$). So the probability of successfully decoding the original data is:

$$\Pr[X \geq n] \;=\; \sum_{i=n}^{n+k} \Pr[X=i] \;=\; \sum_{i=n}^{n+k} \binom{n+k}{i} (1-p)^i p^{n+k-i}.$$

Given fixed n and p, we can choose k such that this probability is no less than, say, 0.99.

Expectation

The distribution of a r.v. contains *all* the probabilistic information about the r.v. In most applications, however, the complete distribution of a r.v. is very hard to calculate. For example, consider the homework permutation example with $n = 20$. In principle, we'd have to enumerate $20! \approx 2.4 \times 10^{18}$ sample points,

compute the value of X at each one, and count the number of points at which X takes on each of its possible values! (though in practice we could streamline this calculation a bit). Moreover, even when we can compute the complete distribution of a r.v., it is often not very informative.

For these reasons, we seek to *compress* the distribution into a more compact, convenient form that is also easier to compute. The most widely used such form is the *expectation* (or *mean* or *average*) of the r.v.

Definition 16.3 (Expectation)**.** *The expectation of a discrete random variable X is defined as*

$$\mathrm{E}(X) = \sum_{a \in \mathscr{A}} a \times \Pr[X = a],$$

where the sum is over all possible values taken by the r.v.

For our simpler permutation example with only 3 students, the expectation is

$$\mathrm{E}(X) = \left(0 \times \frac{1}{3}\right) + \left(1 \times \frac{1}{2}\right) + \left(3 \times \frac{1}{6}\right) = 0 + \frac{1}{2} + \frac{1}{2} = 1.$$

That is, the expected number of fixed points in a permutation of three items is exactly 1.

The expectation can be seen in some sense as a "typical" value of the r.v. (though note that it may not actually be a value that the r.v. ever takes on). The question of how typical the expectation is for a given r.v. is a very important one that we shall return to in a later lecture.

Here is a physical interpretation of the expectation of a random variable: imagine carving out a wooden cutout figure of the probability distribution as in Figure 4. Then the expected value of the distribution is the balance point (directly below the center of gravity) of this object.

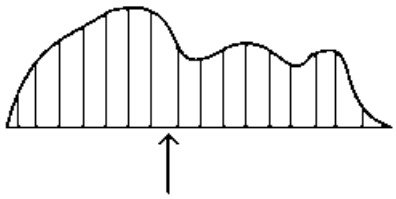


Figure 4: Physical interpretation of expected value as the balance point.

Examples

1. **Single die.** Throw one fair die. Let X be the number that comes up. Then X takes on values $1, 2, \ldots, 6$ each with probability $\frac{1}{6}$, so

$$\mathrm{E}(X) = \frac{1}{6}(1+2+3+4+5+6) = \frac{21}{6} = \frac{7}{2}.$$

   Note that X never actually takes on its expected value $\frac{7}{2}$.

2. **Two dice.** Throw two fair dice. Let X be the sum of their scores. Then the distribution of X is

a	2	3	4	5	6	7	8	9	10	11	12
$\Pr[X = a]$	$\frac{1}{36}$	$\frac{1}{18}$	$\frac{1}{12}$	$\frac{1}{9}$	$\frac{5}{36}$	$\frac{1}{6}$	$\frac{5}{36}$	$\frac{1}{9}$	$\frac{1}{12}$	$\frac{1}{18}$	$\frac{1}{36}$

The expectation is therefore

$$\mathrm{E}(X) = \left(2 \times \frac{1}{36}\right) + \left(3 \times \frac{1}{18}\right) + \left(4 \times \frac{1}{12}\right) + \cdots + \left(12 \times \frac{1}{36}\right) = 7.$$

3. **Roulette.** A roulette wheel is spun (recall that a roulette wheel has 38 slots: the numbers $1, 2, \ldots, 36$, half of which are red and half black, plus 0 and 00, which are green). You bet \$1 on Black. If a black number comes up, you receive your stake plus \$1; otherwise you lose your stake. Let X be your net winnings in one game. Then X can take on the values $+1$ and -1, and $\Pr[X = 1] = \frac{18}{38}$, $\Pr[X = -1] = \frac{20}{38}$. Thus,

$$\mathrm{E}(X) = \left(1 \times \frac{18}{38}\right) + \left(-1 \times \frac{20}{38}\right) = -\frac{1}{19};$$

i.e., you expect to lose about a nickel per game. Notice how the zeros tip the balance in favor of the casino!

Linearity of expectation

So far, we've computed expectations by brute force: i.e., we have written down the whole distribution and then added up the contributions for all possible values of the r.v. The real power of expectations is that in many real-life examples they can be computed much more easily using a simple shortcut. The shortcut is the following:

Theorem 16.1. *For any two random variables X and Y on the same probability space, we have*

$$\mathrm{E}(X+Y) = \mathrm{E}(X) + \mathrm{E}(Y).$$

Also, for any constant c, we have

$$\mathrm{E}(cX) = c\mathrm{E}(X).$$

Proof. To make the proof easier, we will first rewrite the definition of expectation in a more convenient form. Recall from Definition 16.3 that

$$\mathrm{E}(X) = \sum_{a \in \mathscr{A}} a \times \Pr[X = a].$$

Let's look at one term $a \times \Pr[X = a]$ in the above sum. Notice that $\Pr[X = a]$, by definition, is the sum of $\Pr[\omega]$ over those sample points ω for which $X(\omega) = a$. And we know that every sample point $\omega \in \Omega$ is in exactly one of these events $X = a$. This means we can write out the above definition in a more long-winded form as

$$\mathrm{E}(X) = \sum_{\omega \in \Omega} X(\omega) \times \Pr[\omega]. \tag{2}$$

This equivalent definition of expectation will make the present proof much easier (though it is usually less convenient for actual calculations).

Now let's write out $\mathrm{E}(X+Y)$ using equation (2):

$$\begin{aligned}
\mathrm{E}(X+Y) &= \textstyle\sum_{\omega \in \Omega} (X+Y)(\omega) \times \Pr[\omega] \\
&= \textstyle\sum_{\omega \in \Omega} (X(\omega) + Y(\omega)) \times \Pr[\omega] \\
&= \textstyle\sum_{\omega \in \Omega} \Big(X(\omega) \times \Pr[\omega]\Big) + \sum_{\omega \in \Omega} \Big(Y(\omega) \times \Pr[\omega]\Big) \\
&= \mathrm{E}(X) + \mathrm{E}(Y).
\end{aligned}$$

In the last step, we used equation (2) twice.

This completes the proof of the first equality. The proof of the second equality is much simpler and is left as an exercise. □

Theorem 16.1 is very powerful: it says that the expectation of a sum of r.v.'s is the sum of their expectations, with no assumptions about the r.v.'s. We can use Theorem 16.1 to conclude things like $\mathrm{E}(3X-5Y) = 3\mathrm{E}(X)-5\mathrm{E}(Y)$. This property is known as linearity of expectation.

Important caveat: Theorem 16.1 does **not** *say that* $\mathrm{E}(XY) = \mathrm{E}(X)\mathrm{E}(Y)$*, or that* $\mathrm{E}(\frac{1}{X}) = \frac{1}{\mathrm{E}(X)}$*, etc. These claims are not true in general. It is only* sums *and* differences *and* constant multiples *of random variables that behave so nicely.*

Examples

Now let's see some examples of Theorem 16.1 in action.

4. **Two dice again.** Here's a much less painful way of computing $\mathrm{E}(X)$, where X is the sum of the scores of the two dice. Note that $X = X_1 + X_2$, where X_i is the score on die i. We know from example 1 above that $\mathrm{E}(X_1) = \mathrm{E}(X_2) = \frac{7}{2}$. So by Theorem 16.1 we have $\mathrm{E}(X) = \mathrm{E}(X_1) + \mathrm{E}(X_2) = 7$.

5. **More roulette.** Suppose we play the above roulette game not once, but 1000 times. Let X be our expected net winnings. Then $X = X_1 + X_2 + \ldots + X_{1000}$, where X_i is our net winnings in the ith play. We know from earlier that $\mathrm{E}(X_i) = -\frac{1}{19}$ for each i. Therefore, by Theorem 16.1, $\mathrm{E}(X) = \mathrm{E}(X_1) + \mathrm{E}(X_2) + \cdots + \mathrm{E}(X_{1000}) = 1000 \times (-\frac{1}{19}) = -\frac{1000}{19} \approx -53$. So if you play 1000 games, you expect to lose about \$53.

6. **Homeworks.** Let's go back to our homework permutation example with $n = 20$ students. Recall that the r.v. X is the number of students who receive their own homework after shuffling (or equivalently, the number of fixed points). To take advantage of Theorem 16.1, we need to write X as the *sum* of simpler r.v.'s. But since X *counts* the number of times something happens, we can write it as a sum using the following trick:

$$X = X_1 + X_2 + \ldots + X_{20}, \qquad \text{where } X_i = \begin{cases} 1 & \text{if student } i \text{ gets her own hw;} \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

[You should think about this equation for a moment. Remember that all the X's are random variables. What does an equation involving random variables mean? What we mean is that, *at every sample point* ω, we have $X(\omega) = X_1(\omega) + X_2(\omega) + \ldots + X_{20}(\omega)$. Why is this true?]

A $\{0,1\}$-valued random variable such as X_i is called an indicator random variable of the corresponding event (in this case, the event that student i gets her own hw). For indicator r.v.'s, the expectation is particularly easy to calculate. Namely,

$$\mathrm{E}(X_i) = (0 \times \Pr[X_i = 0]) + (1 \times \Pr[X_i = 1]) = \Pr[X_i = 1].$$

But in our case, we have

$$\Pr[X_i = 1] = \Pr[\text{student } i \text{ gets her own hw}] = \frac{1}{20}.$$

Now we can apply Theorem 16.1 to (3), to get

$$\mathrm{E}(X) = \mathrm{E}(X_1) + \mathrm{E}(X_2) + \cdots + \mathrm{E}(X_{20}) = 20 \times \frac{1}{20} = 1.$$

So we see that the expected number of students who get their own homeworks in a class of size 20 is 1. But this is exactly the same answer as we got for a class of size 3! And indeed, we can easily see from the above calculation that we would get $\mathrm{E}(X) = 1$ for *any* class size n: this is because we can write $X = X_1 + X_2 + \ldots + X_n$, and $\mathrm{E}(X_i) = \frac{1}{n}$ for each i.

So the expected number of fixed points in a random permutation of n items is always 1, regardless of n. Amazing, but true.

7. **Coin tosses.** Toss a fair coin 100 times. Let the r.v. X be the number of Heads. As in the previous example, to take advantage of Theorem 16.1 we write

$$X = X_1 + X_2 + \ldots + X_{100},$$

where X_i is the indicator r.v. of the event that the i-th toss is Heads. Since the coin is fair, we have

$$\mathrm{E}(X_i) = \Pr[X_i = 1] = \Pr[i\text{-th toss is Heads}] = \tfrac{1}{2}.$$

Using Theorem 16.1, we therefore get

$$\mathrm{E}(X) = \sum_{i=1}^{100} \frac{1}{2} = 100 \times \tfrac{1}{2} = 50.$$

More generally, the expected number of Heads in n tosses of a fair coin is $\frac{n}{2}$. And in n tosses of a biased coin with Heads probability p, it is np (why?). So the expectation of a binomial r.v. $X \sim Bin(n,p)$ is equal to np. Note that it would have been harder to reach the same conclusion by computing this directly from definition of expectation.

8. **Balls and bins.** Throw m balls into n bins. Let the r.v. X be the number of balls that land in the first bin. Then X behaves exactly like the number of Heads in m tosses of a biased coin, with Heads probability $\frac{1}{n}$ (why?). So from Example 7 we get $\mathrm{E}(X) = \frac{m}{n}$.

   In the special case $m = n$, the expected number of balls in any bin is 1. If we wanted to compute this directly from the distribution of X, we'd get into a messy calculation involving binomial coefficients.

   Here's another example on the same sample space. Let the r.v. Y be the number of empty bins. The distribution of Y is horrible to contemplate: to get a feel for this, you might like to write it down for $m = n = 3$ (3 balls, 3 bins). However, computing the expectation $\mathrm{E}(Y)$ is easy using Theorem 16.1. As usual, let's write

$$Y = Y_1 + Y_2 + \cdots + Y_n, \tag{4}$$

   where Y_i is the indicator r.v. of the event "bin i is empty". Again as usual, the expectation of Y_i is easy:

$$\mathrm{E}(Y_i) = \Pr[Y_i = 1] = \Pr[\text{bin } i \text{ is empty}] = \left(1 - \frac{1}{n}\right)^m;$$

   recall that we computed this probability (quite easily) in an earlier lecture. Applying Theorem 16.1 to (4) we therefore have

$$\mathrm{E}(Y) = \sum_{i=1}^{n} \mathrm{E}(Y_i) = n\left(1 - \frac{1}{n}\right)^m,$$

a very simple formula, very easily derived.

Let's see how it behaves in the special case $m = n$ (same number of balls as bins). In this case we get $\mathrm{E}(Y) = n(1-\frac{1}{n})^n$. Now the quantity $(1-\frac{1}{n})^n$ can be approximated (for large enough values of n) by the number $\frac{1}{\mathrm{e}}$.[2] So we see that

$$\mathrm{E}(Y) \to \frac{n}{\mathrm{e}} \approx 0.368n \quad \text{as } n \to \infty.$$

The bottom line is that, if we throw (say) 1000 balls into 1000 bins, the expected number of empty bins is about 368.

[2]More generally, it is a standard fact that for any constant c,

$$(1+\tfrac{c}{n})^n \to \mathrm{e}^c \quad \text{as } n \to \infty.$$

We just used this fact in the special case $c = -1$. The approximation is actually very good even for quite small values of n — try it yourself! E.g., for $n = 20$ we already get $(1-\frac{1}{n})^n = 0.358$, which is very close to $\frac{1}{\mathrm{e}} = 0.367\ldots$. The approximation gets better and better for larger n.

CS 70 Discrete Mathematics and Probability Theory
Spring 2016 Rao and Walrand JEAN WALRAND - Probability Review

The objective of these notes is to enable you to check your understanding and knowledge of the probability concepts studied in this course. These notes are not meant to be self-contained. They are mostly a check-list. The lecture slides and lecture notes contain the details. You should try to invent problems and see if you can solve them.

1 Probability Space

Review

The *sample space* Ω is the set of possible *outcomes* of some random experiment. The experiment selects *one* outcome (only one!) $\omega \in \Omega$. The *probability* that it selects ω is $Pr[\omega]$. For now (until Section 8), Ω is either finite or countable (i.e., its elements can be numbered $\omega_1, \omega_2, \ldots$). For each $\omega \in \Omega$, one has $Pr[\omega] \geq 0$ and $\sum_\omega Pr[\omega] = 1$. A finite probability space is said to be *uniform* when all the outcomes have the same probability $1/|\Omega|$.

An *event* is a subset A of Ω, i.e., a set of outcomes. One defines the *probability of the event A* as $Pr[A] = \sum_{\omega \in A} Pr[\omega]$. Note that if $A \cap B = \emptyset$, then $Pr[A \cup B] = Pr[A] + Pr[B]$. By induction, if the events A_m are pairwise disjoint, then $Pr[A_1 \cup \cdots \cup A_n] = Pr[A_1] + \cdots + Pr[A_n]$ for all finite n. One can also show that this identity holds even for infinite n, i.e., $Pr[\cup_{m \geq 1} A_m] = \sum_{m \geq 1} Pr[A_m]$ if the events A_m are pairwise disjoint. One says that probability is *countably additive*.

By induction, one also sees that for arbitrary events A_m one has $Pr[A_1 \cup \cdots \cup A_n] \leq Pr[A_1] + \cdots + Pr[A_n]$. This inequality is called the *union bound*. Also, for any two events A and B one has $Pr[A \cup B] = Pr[A] + Pr[B] - Pr[A \cap B]$. This is called the *inclusion-exclusion* identity.

Symmetry is a powerful argument to determine that two events have the same probability. For instance, say that there is a bag with 100 red balls and 200 blue balls. You pick five balls without replacement. The probability that the fifth ball is red is $1/3$. Indeed, this is the same as the probability that the first ball is red. To see this, think of arranging the 300 balls in a random order (permutation), so that all the permutations are equally likely. They remain so if you interchange the first and fifth ball. As another example, you deal five cards from a well-shuffled 52-card deck. The probability that the third card is red is $1/2$. The probability that the fourth card is an ace is $4/52 = 1/13$. Of course, the likelihood that the fourth card is an ace depends on the first three cards. Say that there is only one specific chocolate that you like in a box that you share with some friends by everyone in turn picking one chocolate randomly uniformly and without replacement. If you are the last one to pick, you might think that you are less likely to get your favorite chocolate than if you are first. Not so, by symmetry.

Examples

1. $\Omega = \{1,2,3,4\}$ be a uniform probability space. For $A = \{1,2,3\}$ and $B = \{3,4\}$, one finds that

   $$Pr[A] = 3/4, Pr[B] = 1/2, Pr[A \cap B] = 1/4, Pr[A \setminus B] = 1/2, Pr[\bar{A}] = 1/4, Pr[A \cup B] = 1, Pr[A \Delta B] = 3/4.$$

2. $\Omega = \{HH, HT, TH, TT\}$ be a uniform probability space. This probability space describes the random experiment "flipping a fair coin twice." Note that the outcome of the experiment is one of the elements of Ω. It would be fundamentally wrong to say that this experiment is described by $\Omega = \{H, T\}$ where one picks two elements of Ω when one performs the experiment. It is wrong because we want to describe how the coin flips are related. For instance, if we glue the two coins together so that they both land on the same face, then the probability space becomes the uniform space $\{HH, TT\}$. We lose this relationship if we think of $\Omega = \{H, T\}$ and we select two outcomes. Hence, it is essential that ω describes the full outcome of the complete experiment, i.e., each ω describes the two coin flips. *Make sure you understand this point.* What would the probability space be if you flipped the fair coin 100 times?

3. $\Omega = \{H, T\}$ with $Pr[H] = 0.6$ and $Pr[T] = 0.4$. This probability space describes the random experiment "flipping a biased coin once."

4. $\Omega = \{1, 2, \ldots\}$ with $Pr[n] = (1-p)^{n-1}p$ for $n \geq 1$. This probability space describes the random experiment "flipping a biased coin until it yields the first H."

2 Conditional Probability and Independence

Review

For two events A and B in Ω, one defines the *conditional probability of A given B* as $Pr[A|B] := Pr[A \cap B]/Pr[B]$. We say that A and B are independent if $Pr[A \cap B] = Pr[A]Pr[B]$. Three events A, B, C are mutually independent if they are independent two by two and if, in addition, $Pr[A \cap B \cap C] = Pr[A]Pr[B]Pr[C]$. More generally, the events $\{A_i, i \in I\}$ are mutually independent if

$$Pr[A_{i_1} \cap A_{i_2} \cap \cdots \cap A_{i_n}] = Pr[A_{i_1}] \times \cdots \times Pr[A_{i_n}]$$

for all $n \geq 2$ and any $\{i_1, \ldots, i_n\} \subset I$. We say that the events A and B are positively (resp., negatively) correlated if $Pr[A \cap B] > Pr[A]Pr[B]$ (resp., $Pr[A \cap B] < Pr[A]Pr[B]$). Thus, if A and B are positively correlated, then $Pr[B|A] > Pr[B]$. Note that this does not imply that A causes B. For instance, one also has $Pr[A|B] > Pr[A]$, so is it A that causes B or B that causes A? In fact, it may be neither: the events could have a common cause, like people who drive a Tesla being more likely to own a vacation home, and vice-versa.

Examples

1. Let $\Omega = \{1, 2, 3, 4\}$ be a uniform probability space. Let also $A = \{1, 2, 3\}$ and $B = \{3, 4\}$. Then $Pr[A|B] = 1/2, Pr[B|A] = 1/3$. Here, A and B are negatively correlated.

2. Roll a balanced six-sided die. The probability that the outcome is 6 given that it is larger than 4 is $1/2$.

3. A couple has two kids, at least one of which is a girl. The probability that they have two girls is $Pr[GG]/Pr[GB, BG, GG] = 1/3$. Perhaps surprisingly, it is not $1/2$.

4. Roll a balanced six-sided die twice. Let A be the event that the first roll yields m for some $m \in \{1, \ldots, 6\}$ and B the event that the second roll yields n for some n in $\{1, \ldots, 6\}$. Then A and B are independent. We say that the two rolls are independent. (See Section 4.)

5. Let $\Omega = \{1,\ldots,8\}$ be a uniform probability space. Let $A = \{1,2,3,4\}, B = \{4,5\}, C = \{1,2,5,6\}$. Then A, B, C are pairwise independent, not mutually. Note that A, B, C would not be pairwise independent if the uniform probability space were $\{1,\ldots,6\}$.

6. Let $\Omega = \{1,\ldots,8\}$ be a uniform probability space. Then $A = \{1,2,3,4\}, B = \{2,4,6,8\}, C = \{2,3,6,7\}$ are mutually independent.

7. Let $\{A_i, i \in I\}$ be mutually independent. Let also J and K be disjoint finite subsets of I. Define E to be an event obtained by performing set operations on the events $\{A_i, i \in J\}$ and F an event obtained by performing set operations on the events $\{A_i, i \in K\}$. For instance, $E = (A_1 \cup A_2) \setminus (A_3 \cap A_4)$ and $F = (A_5 \Delta A_6) \setminus A_7$. Then E and F are independent. Here is a general proof, for arbitrary E and F. First look at the case of three events A_1, A_2, A_3. When you draw the Venn diagram, you find that there 8 disjoint sets that make up Ω: $B_1 = A_1 \cap A_2 \cap A_3, B_2 = A_1 \cap A_2^c \cap A_3^c, \ldots, B_8 = A_1^c \cap A_2^c \cap A_3^c$. These events are of the form $D_1 \cap D_2 \cap D_3$ where each D_i is either A_i or A_i^c. Any event that you can create by performing set operations on A_1, A_2, A_3 is a union of some of the eight sets B_j. More generally, define the *atoms* B_n of $\{A_i, i \in J\}$ to be all the events of the form $B_n = \cap_{i \in J} D_i$ where each D_i is either A_i or A_i^c. Similarly, define the atoms C_m of $\{A_i, i \in K\}$. These atoms B_n are pairwise disjoint. Similarly, the atoms C_m are pairwise disjoint. Also, each B_n is independent of every C_m. Moreover, E is a union of atoms B_n and F is a union of atoms C_m. Thus, $Pr[E \cap F] = Pr[\cup_{m,n}(B_n \cap C_m)] = \sum_{m,n} Pr[B_n \cap C_m] = \sum_{m,n} Pr[B_n]Pr[C_m] = (\sum_n Pr[B_n])(\sum_m Pr[C_m]) = Pr[E]Pr[F]$.

3 Bayes' Rule

Review

The setup is that Ω is a probability space, $\{A_1,\ldots,A_n\}$ are a partition of Ω and B is some event. Then

$$Pr[A_m|B] = \frac{Pr[A_m]Pr[B|A_m]}{\sum_{k=1}^n Pr[A_k]Pr[B|A_k]}, m = 1,\ldots,n.$$

The point of this formula is as follows. We think of the A_m as possible pairwise exclusive *circumstances* under which the *symptom* B can occur. One knows the *prior* probability $Pr[A_k]$ of every circumstance. One also knows the conditional probability $Pr[B|A_k]$ that B occurs under circumstance A_k . Bayes' Rule tells us how to compute the likelihood $Pr[A_k|B]$ that circumstance A_k is in effect given that B occurs. Think of A_k as a disease and B as a symptom such as a fever, or a suspicious X-ray reading, or some abnormal value for a test. One calls $Pr[A_m|B]$ the *posterior* probability of A_m given B. Thus, Bayes' Rule computes the posterior probabilities $Pr[A_m|B]$ given the prior probabilities $Pr[A_m]$ and the conditional probabilities $Pr[B|A_m]$.

The event A_m with the maximum value of $Pr[A_m|B]$ is said to be the *Maximum A Posteriori* (MAP) estimate of the circumstance given the symptom, i.e., the most likely circumstance A_m given that B occurs. The event A_m with the maximum value of $Pr[B|A_m]$ is said to be the *Maximum Likelihood Estimate* (MLE) of the circumstance given the symptom. The MLE and the MAP coincide if the priors are uniform, i.e., if $Pr[A_m] = 1/n$ for $m = 1,\ldots,n$. Otherwise, they generally differ. Thus, the MAP is the most likely A_m given B while the MLE is the A_m that makes B most likely. For instance, Ebola is the MLE circumstance of diarrhea whereas food poisoning may be the MAP.

Examples

1. A coin is equally likely to be fair or biased with $Pr[H] = 0.7$. You flip it once and get H. The posterior

probability that the coin is fair is

$$\frac{0.5 \times 0.5}{0.5 \times 0.5 + 0.5 \times 0.7} \approx 0.42.$$

Thus, the coin is a bit more likely to be biased since it produced one H. The probability that the next coin flip yields H is then

$$0.42 \times 0.5 + 0.58 \times 0.7 \approx 0.62.$$

If the first flip yields T, then you can verify that the likelihood that the second flip yields H is 0.575.

2. A coin is fair with probability 0.7, otherwise it is such that $Pr[H] = 0.6$. You flip the coin ten times and get 8 heads. The posterior probability that the coin is fair is

$$\frac{0.7\binom{10}{8}(0.5)^8(0.5)^2}{0.7\binom{10}{8}(0.5)^8(0.5)^2 + 0.3\binom{10}{8}(0.6)^8(0.4)^2} \approx 0.1.$$

Thus, it is much more likely that the coin is biased since it produced so many heads. Consequently, the conditional probability that flip 11 yields heads is approximately $0.1 \times 0.5 + 0.9 \times 0.6 \approx 0.59$.

3. A coin is fair with probability 0.5, otherwise it is such that $Pr[H] = 0.6$. You flip the coin repeatedly and it takes 10 flips until the first H. The posterior probability that it is fair is

$$\frac{0.5 \times (0.5)^9(0.5)}{0.5 \times (0.5)^9(0.5) + 0.5 \times (0.4)^9(0.6)} \approx 0.86.$$

Thus, the conditional probability that the coin is fair is much larger than the prior probability 0.5 because it took so long to get the first H. Consequently, the probability that the next flip, i.e., flip 11, yields heads is approximately $0.86 \times 0.5 + 0.14 \times 0.6 \approx 0.51$.

4. There are two envelopes. The first one contains five checks in the amounts of $\{1,2,5,5,6\}$ and the second contains four checks in the amounts $\{2,5,6,8\}$. You pick an envelope at random and then pick a check at random. Given that the check you got is in the amount of 5, the posterior probability that you picked it from the first envelope is

$$\frac{0.5 \times (2/5)}{0.5 \times (2/5) + 0.5 \times (1/4)} \approx 0.62.$$

Thus, if you are offered to either keep the envelope you selected or to switch to the other envelope, you should switch because the second envelope contains more money.

4 Random Variables and Expectation

Review

A *random variable* is a *real-valued function of the outcome of a random experiment*. Thus, there is a probability space Ω with $Pr[\omega]$ defined for each $\omega \in \Omega$. This probability space defines the random experiment. A random variable X is a function $X : \Omega \to \Re$ that assigns a real number $X(\omega)$ to each outcome $\omega \in \Omega$. For $A \subset \Re$, one defines $Pr[X \in A] := Pr[X^{-1}(A)]$ where $X^{-1}(A) := \{\omega \in \Omega \mid X(\omega) \in A\}$ is the *inverse image* of A under the function X. Similarly, $Pr[X = a] := Pr[X^{-1}(\{a\})]$. The *distribution* of a random variable X is the set of its possible values and their probability. This may seem abstract, so here is a concrete example. Say that a bag contains 100 balls. Among those, 23 are marked with the value 5 and the others are

marked with different values. Let ω be the ball you pick and $X(\omega)$ the value marked on the ball. Then $Pr[X=5] = Pr[X^{-1}(5)] = 23/100$. Here, $X^{-1}(5)$ is the set of 23 balls marked with the value 5.

If X, Y are two random variables on Ω and $g : \Re^2 \to \Re$ is a function, then $g(X,Y)$ is a new random variable that assigns the real number $g(X(\omega), Y(\omega))$ to $\omega \in \Omega$.

The expected value of X is $E[X] := \sum_x xPr[X=x] = \sum_\omega X(\omega)Pr[\omega]$. Thus, $E[g(X,Y)] = \sum_\omega g(X(\omega), Y(\omega))Pr[\omega]$. Expectation is linear: $E[aX+bY] = aE[X] + bE[Y]$.

Given an event A, one defines the *conditional expectation* of X given A as $E[X|A] := \sum_x xPr[X=x|A] = \sum_\omega X(\omega)Pr[\omega|A]$. Assume that $\{A_1, \ldots, A_n\}$ is a partition of Ω. Then $E[X] = \sum_m E[X|A_m]Pr[A_m]$. In particular, $E[X] = \sum_y E[X|Y=y]Pr[Y=y]$. If we define the *conditional expectation* of X given Y as $E[X|Y] := g(Y)$ where $g(y) := E[X|Y=y]$, then we see that $E[X] = E[E[X|Y]]$.

One also finds that $E[Xg(Y)|Y] = g(Y)E[X|Y]$, so that $E[(X - E[X|Y])g(Y)] = 0$, an identity that shows that the *estimation error* $X - E[X|Y]$ is orthogonal to any function $g(Y)$. This *projection property* implies that $E[(X-h(Y))^2]$ is minimized by choosing $h(Y) = E[X|Y]$. We say that the conditional expectation $E[X|Y]$ is the *Minimum Mean Squares Estimate* (MMSE) of X given Y. Another way to appreciate this property is to note that $E[(X-h(Y))^2] = \sum_y Pr[Y=y]E[(X-h(y))^2|Y=y]$ and that, for each y, $E[(X-h(y))^2|Y=y]$ is minimized by choosing $h(y) = E[X|Y=y]$, as we see by setting to zero the derivative with respect to a of $E[(X-a)^2|Y=y] = E[X^2|Y=y] - 2aE[X|Y=y] + 2a^2$.

We say that the random variables X and Y are independent if (and only if) $Pr[X=x, Y=y] = Pr[X=x]Pr[Y=y]$ for all x, y. Equivalently, they are independent if and only if $Pr[X \in A, Y \in B] = Pr[X \in A]Pr[Y \in B]$ for all $A, B \subset \Re$. If X and Y are independent, then $g(X)$ and $h(Y)$ are independent for any functions g and h. More generally, the random variables $\{X_i, i \in I\}$ are *mutually independent* if $Pr[X_{i_1} \in A_1, \ldots, X_{i_n} \in A_n] = Pr[X_{i_1} \in A_1] \times \cdots \times Pr[X_{i_n} \in A_n]$ for all finite n, all $i_1, \ldots, i_n \in I$ and all sets $A_1, \ldots, A_n$ in $\Re$. If the random variables X_i are mutually independent, then $E[X_{i_1}X_{i_2}\cdots X_{i_n}] = E[X_{i_1}]\cdots E[X_{i_n}]$. The converse is not true.

Examples

1. Flip a biased coin with $Pr[H] = p$. Let $X = 1$ if the outcome is H and 0 otherwise. Then the distribution of X is called Bernoulli with parameter p and one writes $X = B(p)$. Thus, $E[X] = p$ and $E[X^2] = E[X] = p$.

2. Let X_m for $m = 1, \ldots, n$ be i.i.d. $B(p)$. Then $X = X_1 + \ldots + X_n = B(n,p)$. Thus, $E[X] = np$.

3. Roll a balanced six-sided die once and let X be the number of pips. Then $E[X] = \sum_{m=1}^6 m(1/6) = 3.5$.

4. Roll a balanced six-sided die twice. Then $\Omega = \{1, \ldots, 6\}^2$ with $Pr[\omega] = 1/36$ for all $\omega \in \Omega$. Let $X((a,b)) = a+b$ for $(a,b) \in \Omega$. Then $E[X] = 2 \times 3.5 = 7$, by linearity of expectation.

5. Same experiment as above. Let $Y((a,b)) = \min\{a,b\}$ and $Z((a,b)) = \max\{a,b\}$ for $(a,b) \in \Omega$. Then (it may be useful to draw a picture) $Pr[Y=1] = 11/36, Pr[Y=2] = 9/36, Pr[Y=3] = 7/36, Pr[Y=4] = 5/36, Pr[Y=5] = 3/36, Pr[Y=6] = 1/36$. Also, $Pr[Z=m] = Pr[Y=6-m]$, by symmetry. Hence, $E[Y] = 1(11/36) + 2(9/36) + \cdots + 6(1/36) = 91/36 \approx 2.52$. Also, $Y+Z$ is the total number of pips on the two rolls, so that $E[Y+Z] = 7$ and it follows that $E[Z] = 7 - E[Y] \approx 4.48$.

6. Pick a number uniformly in $\{1, \ldots, n\}$. Then $\Omega = \{1, \ldots, n\}$ with $Pr[m] = 1/n$ for $m \in \Omega$. Define $X(m) = 2 + 3m + 4m^2$.

7. Let $A \subset \Omega$ and define $X(\omega) = 1\{\omega \in A\}$ for $\omega \in \Omega$. The random variable X is called the *indicator* of the event A. Sometimes we write it as $1_A(\omega)$.

8. Flip n times a biased coin with $Pr[H] = p$. Then $\Omega = \{H,T\}^n$ and $Pr[\omega] = p^m(1-p)^{n-m}$ if ω has m heads. Define $X(\omega)$ to be the number of heads in ω. Then $Pr[X = m] = \binom{n}{m} p^m(1-p)^{n-m}$. This is the *binomial distribution* and we write it as $B(n,p)$.

9. Flip a biased coin with $Pr[H] = p$ until you get a first H. Then $\Omega = \{1,2,\ldots\}$ and $Pr[n] = (1-p)^{n-1}p$ for $n \geq 1$. Let $X(n) = n$. The distribution of X is called the *Geometric distribution with parameter p*. We designate it by $G(p)$.

10. A monkey is typing away on a keyboard that has only the 26 letters. After he types 10^{12} letters, let X be the number of times that the word 'walrand' appears. Then $E[X] = (10^{12}-7)(26)^{-7} \approx 124$. Indeed, $X = X_1 + \cdots + X_n$ with $n = 10^{12} - 7$ where X_m is the indicator of the event that the word 'walrand' appears starting with the m-th letter. One has $E[X_m] = (26)^{-7}$ and $E[X] = nE[X_1]$, by linearity of expectation.

11. An elevator loads n people on the ground floor of a building with $k+1$ floors (including the ground floor 0). Each person chooses one of the k other floors independently, with equal probabilities. The probability that no one chooses a particular floor is then $(1-1/k)^n$. Consequently, if X_m is the indicator that someone chose floor m, one sees that $E[X_m] = 1-(1-1/k)^n$. Let $X = X_1 + \cdots + X_k$ be the number of different floors that people picked. Then, by linearity of expectation, $E[X] = kE[X_1] = k[1-(1-1/k)^n]$.

12. Let $X = G(p)$, so that $Pr[X = n] = (1-p)^{n-1}p$ for $n \geq 1$. Then $E[X] = 1/p$.

13. Let $X = G(p)$ and $Y = G(q)$ be independent. Then $\min\{X,Y\} = G(r)$ with $r = 1-(1-p)(1-q)$.

14. We say that X is Poisson with parameter $\lambda > 0$ if $Pr[X = n] = (\lambda^n)/(n!)\exp\{-\lambda\}$ for $n \geq 0$. We write $X = P(\lambda)$. Then $E[X] = \lambda$.

15. Let $X = P(\lambda)$ and $Y = P(\mu)$ be independent. Then $X + Y = P(\lambda + \mu)$.

16. Flip a coin. With probability p, the outcome is H and one defines $Z = X$; otherwise, one defines $Z = Y$. Here, X and Y are two random variables that are independent of the coin flip. Find $E[Z]$ and $var(Z)$ in terms of p and the mean and variance of X and Y.

17. Assume that $E[X_{n+1}|X_n] = a + bX_n$ for $n \geq 1$. Taking expectation, we get $E[X_{n+1}] = a + bE[X_n]$. Hence, $E[X_2] = a + bE[X_1], E[X_3] = a + bE[X_2] = a + b(a + bE[X_1]) = a(1+b) + b^2E[X_1]$. Continuing in this way, we find that $E[X_n] = a(1 + b + \cdots + b^{n-2}) + b^{n-1}E[X_1] = a(1-b^{n-1})/(1-b) + b^{n-1}E[X_1]$.

18. Diluting. There is a bin with 100 red balls. At step 1, we pick a ball from the bin and replace it with a blue ball. We now have $X_1 = 99$ red balls and 1 blue ball. We continue in this way and let X_n be the number of red balls in the bin after n steps. At step $n+1$, the likelihood that we pick a red ball is $X_n/100$. Hence, one has $E[X_{n+1}|X_n] = X_n - X_n/100 = 0.99X_n$. Hence, $E[X_n] = (0.99)^n$ for $n \geq 1$.

19. Mixing. There is a bin with 100 red balls and one with 100 blue balls. At each step, one picks a ball from each bin and puts it in the other bin. Let X_n be the number of red balls in the first bin at the end of n steps. Then $E[X_{n+1}|X_n] = X_n - (X_n/100)(X_n/100) + (1-X_n/100)(1-X_n/100) = 1 + bX_n$ with $b = 49/50$. Hence, $E[X_n] = (1-b^{n-1})(1-b) + b^{n-1}99$.

20. Going Viral. Assume everyone on Tweeter has a random number of friends that has mean μ. You tweet a rumor to those friends. Each of them retweets independently with probability p to each of their friends, and so on. Let X_n be the number of people who retweet the rumor at step n. Given $X_n = k$ and the number of friends $Y_1, \ldots, Y_k$ of these k people, we see that $X_{n+1} = B(Y_1 + \cdots + Y_k, p)$.

Thus, $E[Xn+1|X_n = k, Y_1, \ldots, Y_k] = p(Y_1 + \cdots + Y_k)$. Hence, $E[X_{n+1}|X_n = k] = E[p(Y_1 + \cdots + Y_k)] = pk\mu = p\mu X_n$. Consequently, $E[X_{n+1}] = p\mu E[X_n]$ and we conclude that $E[X_n] = (p\mu)^{n-1}$ for $n \geq 1$ (because $X_1 = 1$). If $X = X_1 + X_2 + \cdots$, we see that $E[X] < \infty$ if and only if $p\mu < 1$.

5 Covariance, Variance, Linear Regression and Estimation

Review

The *covariance* of two random variables X and Y is defined as $cov(X,Y) = E[XY] - E[X]E[Y]$. The random variables X and Y are said to be *uncorrelated* if $cov(X,Y) = 0$, *positively correlated* if $cov(X,Y) > 0$ and *negatively correlated* if $cov(X,Y) < 0$. Note that if X and Y are independent, then $cov(X,Y) = 0$. The converse is not true. The *variance* of a random variable is defined as $var(X) = E[X^2] - E[X]^2$. One has $var(X+Y) = var(X) + var(Y) + 2cov(X,Y)$. In particular, if X and Y are independent, then $var(X+Y) = var(X) + var(Y)$. Also, $var(a+bX) = b^2 var(X)$ and $cov(aX+bY, cV+dW) = ac.cov(X,V) + ad.cov(X,W) + bc.cov(Y,V) + bd.cov(Y,W)$.

If $f : \Re \to [0,\infty)$ is nondecreasing and $f(a) > 0$, then $Pr[X \geq a] \leq E[f(X)]/f(a)$. This is Markov's inequality. Chebyshev's inequality states that $Pr[|X - E[X]| \geq a] \leq var[X]/a^2$.

The linear function of X that minimizes $E[(Y - a - bX)^2]$ is $L[Y|X] := E[Y] + [cov(X,Y)/var(X)](X - E[X])$. We call this function the LLSE (linear least squares estimate) of Y given X. In particular, if (X,Y) is distributed uniformly in the set $\{(X_m, Y_m), m = 1, \ldots, n\}$, then $L[Y|X]$ is called the *linear regression* of Y over X. Note that this LR is non-Bayesian: it does not assume a prior distribution of (X,Y) but uses only the observed sample values.

Let $\{X_m, m \geq 1\}$ be i.i.d. with mean μ and variance σ^2 and $A_n := (X_1 + \cdots + X_n)/n$. Then, Chebyshev implies that $Pr[|A_n - \mu| \geq \varepsilon] \leq \sigma^2/(n\varepsilon^2) = \sigma^2/(n\varepsilon^2)$. Thus, for all $\varepsilon > 0$ one has $Pr[|A_n - \mu| \geq \varepsilon] \to 0$ as $n \to \infty$. This result is called the *weak law of large numbers* (WLLN). Using the same inequality and choosing ε so that $n\varepsilon^2 = 20\sigma^2$, i.e., $\varepsilon = 4.5\sigma/\sqrt{n}$, we find that $Pr[|A_n - \mu| \geq 4.5\sigma/\sqrt{n}] \leq 5\%$, which shows that $[A_n - 4.5\sigma/\sqrt{n}, A_n + 4.5\sigma/\sqrt{n}]$ is a 95%-confidence interval for μ. Using the *Central Limit Theorem*, we will be able to replace 4.5 by 2 in the confidence interval (see Section 9).

Examples

1. Let $X = B(p)$. Then $var(X) = E[X^2] - E[X]^2 = E[X] - E[X]^2 = p(1-p) \leq 1/4$.

2. Let $X = B(n,p)$. Then we can write $X = X_1 + \cdots + X_n$ where the X_m are i.i.d. $B(p)$, so that $var(X) = np(1-p)$.

3. Let Ω be the uniform probability space $\{1, \ldots, 6\}$ and let X takes the values $\{0,0,1,1,2,2\}$ and Y the values $\{0,3,6,12,3,0\}$, respectively for the different values of ω. Thus, $X(1) = X(2) = 0$ and $Y(1) = 0, Y(1) = 3$, etc. Then $E[X] = (0+0+1+1+2+2)/6 = 1, E[Y] = (0+3+6+12+3+0)/6 = 4, E[XY] = (0+0+6+12+6+0)/6 = 4$. Hence, $cov(X,Y) = 0$ and $L[Y|X] = E[Y] = 4$. Also, $Pr[Y = 0|X = 0] = 1/2$ and $Pr[Y = 3|X = 0] = 1/2$, so that $E[Y|X = 0] = 3/2$. Similarly, $E[Y|X = 1] = 9$ and $E[Y|X = 2] = 3/2$. We can write $E[Y|X] = (3/4)(X-1)(X-2) - 9(X-0)(X-2) + (3/4)(X-0)(X-1) = -(15/2)X^2 + 15X + 3/2$ since a polynomial that goes through the point (x_m, y_m) for $m = 1, \ldots, n$ can be written as $\sum_{m=1}^{n} y_m \Pi_{k \neq m}[(x - x_k)/(x_m - x_k)]$.

4. Let X, Y, Z be i.i.d. with mean 0 and variance 1. Then $L[aX + bY + cZ|dX + eY + fZ] = [(ad + be + cf)/(d^2 + e^2 + f^2)](dX + eY + fZ)$.

5. Let $\Omega = \{1,2,3,4\}$ be a uniform probability space. Let also $A = \{1,2\}, B = \{1,3\}, C = \{1,4\}$ and $X = 1_A, Y = 1_B, Z = 1_C$. Then, X, Y, Z are pairwise independent, but not mutually independent. They are also identically distributed like $B(0.5)$. Note that $E[XYZ] = 1/4 \neq E[X]E[Y]E[Z]$. Thus, when we write 'i.i.d.', we mean *mutually independent* and identically distributed, like three coin flips, for instance.

6. Let $\{X_1, \ldots, X_n\}$ be pairwise independent. Then $var(X_1 + \cdots + X_n) = var(X_1) + \cdots + var(X_n)$.

7. Let $X = B(n,p)$. Then $X = X_1 + \cdots + X_n$ where the X_m are i.i.d. $B(p)$. Hence, $var(X) = n.var(X_1) = np(1-p)$.

8. Let $X = P(\lambda)$. Then $E[X(X-1)] = \sum_{n\geq 0} n(n-1)\lambda^n \exp\{-\lambda\}/(n!) = \lambda^2 \sum_{n\geq 2} \lambda^{n-2} \exp\{-\lambda\}/[(n-2)!] = \lambda^2 \sum_{n\geq 0} \lambda^n \exp\{-\lambda\}/(n!) = \lambda^2$. Thus, $E[X^2] - E[X] = \lambda^2$, so that $E[X^2] = \lambda^2 + E[X] = \lambda^2 + \lambda$ and $var(X) = E[X^2] - E[X]^2 = \lambda$.

9. Let $X = G(p)$. Then $X = 1 + ZY$ where $Y = G(p)$ and $Z = B(1-p)$ are independent. Thus, $E[X^2] = E[1 + 2ZY + Z^2Y^2] = 1 + 2(1-p)E[Y] + (1-p)E[Y^2] = 1 + 2(1-p)/p + (1-p)E[X^2]$. Hence, $E[X^2] = [1 + 2(1-p)/p]/p = (2-p)/p^2$. Consequently, $var(X) = E[X^2] - E[X]^2 = (2-p)/p^2 - 1/p^2 = (1-p)/p^2$.

6 Collisions and Collecting

Review

Say that there are M different coupons. When you buy a box of cereal, you get coupon m with probability $1/M$ for $m = 1, \ldots, M$. It takes $X_1 = 1$ box to get the first coupon. Then $X_2 = G((M-1)/M)$ to get a new one, then $X_3 = G((M-2)/M)$ to get a new one, and so on. Thus, the average number of boxes required to get all the coupons is $M/M + M/(M-1) + M/(M-1) + \cdots + M = MH(M)$ where $H(M) := 1 + 1/2 + \cdots + 1/M \approx ln(M) + 0.58$. The probability that coupon m has not been seen in n boxes is $[(M-1)/M]^n = (1-1/M)^n \approx \exp\{-n/M\}$. Thus, the average number of coupons seen after n steps is $M[1-(1-1/M)^n] \approx M(1-\exp\{-n/M\})$.

One throws m balls into $n > m$ bins, independently and uniformly at random. The probability that there is no collision after 1 ball is 1, after 2 balls is $(n-1)/n$, after three balls it is $[(n-1)/n] \times [(n-2)/n]$, after four balls it is $[(n-1)/n] \times [(n-2)/n] \times [(n-2)/n]$, and so on. The probability of no collision after m balls is then $\Pi_{k=1}^{m-1}[(n-k)/n] \approx \Pi_{k=1}^{m-1} \exp\{-k/n\} = \exp\{-\sum_{k=1}^{m-1} k/n\} \approx \exp\{-m^2/(2n)\}$. Bin i is still empty after k steps with probability $(1-1/n)^k$. Thus, the expected number of empty bins is $m(1-1/n)^k \approx m\exp\{-k/n\}$. By Markov's inequality, the probability that there is at least one empty bin is at most $m\exp\{-k/n\}$.

Examples

1. Let's use c bits as a CRC for n files. This means that the n files are thrown into $m = 2^c$ bins. The probability of collision is approximately $\exp\{-m^2/(2n)\}$. If we want the probability to be less than $10^{-9} \approx e^{-21}$, we need $m^2/(2n) \geq 21$, i.e., $2^{2c} \geq 42n$, or $2c \geq \log_2(42n) = 5.4 + \log_2(n)$. Hence, $c \geq 2.7 + \log_2(n)/2$. For instance, if $n = 10^{12} \approx 2^{40}$, we need $c \geq 2.7 + 40/2 \approx 23$. The implication is that a 3-byte CRC suffices to sign messages or to detect errors in most applications.

7 Markov Chains

Review

A *Markov chain* on the finite state space $\mathscr{X} = \{1, 2, \ldots, K\}$ is the random sequence $\{X_n, n \geq 0\}$ defined by

$$Pr[X_0 = i_0, X_1 = i_1, \ldots, X_n = i_n] = \pi_0(i_0)P(i_0, i_1) \cdots P(i_{n-1}, i_n)$$

where π_0 is a probability distribution called the *initial distribution* of the Markov chain and P is a $K \times K$ nonnegative matrix whose rows sum to one. In particular, $\pi_n(i) := Pr[X_n = i]$ is such that $\pi_n = \pi_0 P^n$ and $Pr[X_n = j | X_0 = i] = P^n(i, j)$.

A MC is *irreducible* if it can go from any state i to any other state j, possibly in more than one step. It is then *aperiodic* if the return times to some state are coprime. (The return times to any state are then also coprime.)

The distribution π is *invariant* if π solves the *balance equations* $\pi P = \pi$. There is a unique invariant distribution if the Markov chain is irreducible. Moreover, the fraction of time in state i then converges to $\pi(i)$ for all i. The distribution π_n converges to π if the Markov chain is also aperiodic.

The average time $\beta(i)$ to enter a set A of states when starting from state i satisfies the *first step equations*

$$\begin{aligned} \beta(i) &= 1 + \sum_j P(i, j)\beta(j), \forall i \notin A \\ \beta(i) &= 0, \forall i \in A. \end{aligned}$$

The probability $\alpha(i)$ of entering a set A of states before a disjoint set B of states when staring from state i satisfies the *first step equations*

$$\begin{aligned} \alpha(i) &= \sum_j P(i, j)\alpha(j), \forall i \notin A \cup B \\ \alpha(i) &= 1, \forall i \in A \\ \alpha(i) &= 0, \forall i \in B. \end{aligned}$$

Note that we call the equations both for α and β first step equations even though they are not the same. The justification for the terminology is that in both cases one conditions on what happens in the first step of the MC.

Examples

1. The MC on $\{0, 1\}$ with $P(0, 1) = P(1, 0) = a$ and $P(0, 0) = P(1, 1) = 1 - a$ with $a \in [0, 1]$ is irreducible if $a > 0$ and aperiodic if $0 < a < 1$. The invariant distribution is $\pi = [0.5, 0.5]$ when $a > 0$.

2. The MC on $\{0, 1\}$ with $P(0, 1) = a$ and $P(1, 0) = b$ with $a, b \in [0, 1]$ is irreducible if $a, b > 0$ and is then aperiodic if $a \neq 1$ or $b \neq 1$. The invariant distribution is $\pi = [b/(a+b), a/(a+b)]$ when $a, b > 0$.

3. You flip a fair coin until you get two heads in a row. The average number of flips is 6.

4. You roll a balanced six-sided die until the sum of the last two rolls is equal to 8. The average number of rolls is about 8.4.

5. In each step, you get up one rung of a ladder with probability p; otherwise, you fall back to the ground. The average number of step to reach rung 20 is $(p^{-20}-1)/(1-p)$.

6. At each step, you win one dollar with probability p; otherwise you lose it. Starting with n dollars, the probability your fortune reaches $M > n$ before it reaches 0 is $(1-\rho^n)/(1-\rho^M)$ where $\rho = (1-p)p^{-1}$.

8 Continuous Probability

Review

Imagine choosing a real number X uniformly in $[0,1]$. Clearly $Pr[X = x] = 0$ for all $x \in [0,1]$. This would be the same if we chose the number uniformly in $[0,2]$. Thus, the probability of individual outcomes does not describe properly the random experiment. Instead, one starts by defining the probability of events. For choosing uniformly in $[0,1]$, one defines $Pr[[a,b]] = b-a$, for $0 \le a \le b \le 1$. One extents that definition by additivity to any (countable) union of intervals. For instance, $Pr[[0,0.3] \cup [0.7,0.9]] = 0.5$. Thus, for a finite or countable sample space Ω, one starts by defining the probability of each outcome ω and one then defines the probability of an event as the sum of the probabilities of the outcomes it contains. For an uncountable sample space Ω, one starts by defining the probability of its events.

Let $f : \Re \to [0,\infty)$ be a function such that $\int_{-\infty}^{\infty} f(x)dx = 1$. Define a random variable X such that $Pr[x < X < x+\varepsilon] \approx f(x)\varepsilon$ for $\varepsilon \ll 1$. Then $F(x) := Pr[X \le x] = \int_{-\infty}^{x} f(y)dy$. Thus, $f(x)$ is the derivative of $F(x)$. One calls $f(x)$ the *probability density function* (pdf) of X and $F(x)$ the *cumulative distribution function* (cdf) of X. Sometimes one writes $f_X(x) := f(x)$ and $F_X(x) := F(x)$ to specify that the pdf and cdf are those of the random variable X. This is convenient when one deals with more than one random variable. Then $E[X] = \int x f_X(x)dx$ and $E[h(X)] = \int h(x) f_X(x)dx$.

Examples

1. $X = U[a,b]$ iff $f_X(x) = (b-a)^{-1}1\{a < x < b\}$. Consequently, $E[X] = \int_a^b x(b-a)^{-1}dx = (a+b)/2$.

2. $X = Expo(\lambda)$ iff $f_X(x) = \lambda \exp\{-\lambda x\}1\{x \ge 0\}$. Consequently, $E[X] = \int_0^{\infty} x\lambda \exp\{-\lambda x\}dx = -\int_0^{\infty} d\exp\{-\lambda x\}dx$ λ^{-1}.

3. One shoots a dart uniformly in a circle with radius r. Let X be the distance of the dart to the center. Then $F_X(x) = (\pi x^2)/(\pi r^2) = x^2/r^2$ for $0 \le x \le r$. Hence, $f_X(x) = 2x/r^2 1\{0 \le x \le r\}$ and $E[X] = r/3$.

4. Define $Y = a + bX$, for some a and some $b > 0$. Then $f_Y(y)\varepsilon = Pr[y < a+bX < y+\varepsilon] = Pr[\frac{y-a}{b} < X < \frac{y-a}{b} + \frac{\varepsilon}{b}] = f_X(\frac{y-a}{b})\frac{\varepsilon}{b}$. Thus, $f_Y(y) = \frac{1}{b} f_X(\frac{y-a}{b})$.

5. Let X and Y be two independent random variables and $W = \max\{X,Y\}$. Then $F_W(w) = Pr[W \le w] = Pr[X \le w]Pr[Y \le w] = F_X(w)F_Y(w)$.

6. Let X and Y be two independent random variables and $V = \min\{X,Y\}$. Then $1 - F_V(v) = Pr[V > v] = Pr[X > v]Pr[Y > v] = (1-F_X(v))[1-F_Y(v)]$.

7. Let X and Y be two independent random variables and $Z = X+Y$. Then, $f_Z(z)\varepsilon = Pr[z < Z < z+\varepsilon] = \int_{-\infty}^{\infty} Pr[x < X < x+dx, z-x < Y < z-x+\varepsilon] = \int_{-\infty}^{\infty} f_X(x)f_Y(z-x)\varepsilon dx$. Hence, $f_Z(z) = \int_{-\infty}^{\infty} f_X(x)f_Y(z-x)dx$.

8. Assume that with probability p the random variable X has pdf $f_0(x)$ and it has pdf $f_1(x)$ otherwise. Given $X = x$, the probability that it has pdf f_0 is $[pf_0(x)]/[pf_0(x) + (1-p)f_1(x)]$.

9 Gaussian and CLT

Review

A random variable X is $\mathcal{N}(0,1)$ iff its pdf is $f_X(x) = (1/\sqrt{2\pi})\exp\{-x^2/2\}$. One can verify that $E[X] = 0$ and $var(X) = 1$. By definition, the random variable $Y = \mu + \sigma X$ is then $\mathcal{N}(\mu, \sigma^2)$. Then $E[Y] = \mu + \sigma E[X] = \mu$ and $var(Y) = \sigma^2 var(X) = \sigma^2$. Also, one can check that $f_Y(y) = (1/\sqrt{2\pi\sigma^2})\exp\{-(y-\mu)^2/(2\sigma^2)\}$ by Example 4 of the previous section. This is the *bell-shaped* pdf.

If $X = \mathcal{N}(\mu, \sigma^2)$, then $Pr[X > \mu + 2\sigma] \approx 2.5\%$ and $Pr[|X - \mu| > 2\sigma] \approx 5\%$. Also, $Pr[X > \mu + 1.65\sigma] \approx 5\%$ and $Pr[|X - \mu| > 1.65\sigma] \approx 10\%$. Note that Chebyshev shows that $Pr[|X - \mu| > 2\sigma] \leq 1/4 = 25\%$, which is a very loose bound. This is why the CLT (see below) provides smaller confidence intervals than Chebyshev. To get 5% using Chebyshev, we have to write $Pr[|X - \mu| \geq 4.5\sigma] \leq 1/(4.5)^2 \approx 5\%$.

The *Central Limit Theorem* (CLT) states that if the X_n are i.i.d. with mean μ and variance σ^2 and if $A_n = (X_1 + \cdots + X_n)/n$, then $[A_n - \mu]\sqrt{n} \approx \mathcal{N}(0, \sigma^2)$ when $n \gg 1$. Thus, we see that $Pr[|A_n - \mu|\sqrt{n} > 2\sigma] \approx 5\%$, so that $Pr[A_n - 2\sigma/\sqrt{n} \leq \mu \leq A_n + 2\sigma/\sqrt{n}] \approx 95\%$. Hence, $[A_n - 2\sigma/\sqrt{n}, A_n + 2\sigma/\sqrt{n}]$ is a 95%-confidence interval for μ.

One can show that if $X = \mathcal{N}(\mu_1, \sigma_1^2)$ and $Y = \mathcal{N}(\mu_2, \sigma_2^2)$ are independent, then $X + Y = \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$. Intuitively, X and Y are both sums of many small independent random variables, so that $X + Y$ is also, and should therefore be Gaussian. The mean and variance are the sum of those of X and Y.

The Law of Large Numbers implies that if the X_m are i,i.d. with mean μ and variance σ^2, then $A_n = (1/n)\sum_{m=1}^n X_m \approx \mu$ and $s_n^2 := (1/n)\sum_{m=1}^n (X_m - A_n)^2 \approx \sigma^2$. Thus, replacing the standard deviation σ by s_n in the confidence intervals, we see that $[A_n - 2s_n/\sqrt{n}, A_n + 2s_n/\sqrt{n}]$ is a 95%-confidence interval for μ when n is large enough. In practice, one has to be a bit careful because s_n may be a poor estimate of σ when n is not large enough. Use with care and at your own risk!

Examples

1. Let $X = X_1 + \cdots + X_n$ where the X_m are i.i.d. $B(p)$. Let also $A_n = (X_1 + \cdots + X_n)/n$. We saw that $[A_n - 2\sigma/\sqrt{n}, A_n + 2\sigma/\sqrt{n}]$ is a 95%-confidence interval for $E[X_1] = p$. Since $\sigma^2 = var(X_1) = p(1-p) \leq 1/4$, one has $\sigma \leq 1/2$, and it follows that $[A_n - 1/\sqrt{n}, A_n + 1/\sqrt{n}]$ is a 95%-confidence interval for p.

10 Some Important Distributions

1. **Bernoulli with parameter** $p : B(p)$

   $Pr[X = 1] = p; Pr[X = 0] = 1 - p$. Mean $= p$; variance $= p(1-p)$.

2. **Uniform in** $\{1, 2, \ldots, n\} : U[1, \ldots, n]$

   $Pr[X = m] = 1/n, m = 1, \ldots, n$. Mean $= (n+1)/2$; variance $= (n^2 - 1)/12$.

3. **Binomial with parameters** $n, p : B(n, p)$

   $Pr[X = m] = \binom{n}{m} p^m (1-p)^{n-m}, m = 0, \ldots, n$. Mean $= np$; variance $= np(1-p)$.

4. **Geometric with parameter** $p : G(p)$

$Pr[X = n] = (1-p)^{n-1}p, n = 1, 2, \ldots$. Mean $= 1/p$, variance $= (1-p)/p^2$.

5. **Poisson with parameter λ : $P(\lambda)$**

   $Pr[X = n] = (\lambda^n/n!)e^{-\lambda}, n = 0, 1, 2, \ldots$. Mean $= \lambda$, variance $= \lambda$.

6. **Uniform in $[0,1]$: $U[0,1]$.**

   $f_X(x) = 1\{0 \le x \le 1\}$. Mean $= 1/2$, variance $= 1/12$.

7. **Exponential with parameter λ : $Expo(\lambda)$.**

   $f_X(x) = \lambda e^{-\lambda x} 1\{x > 0\}, F_X(x) = [1 - e^{-\lambda x}]1\{x \ge 0\}$. Mean $= \lambda^{-1}$, variance $= \lambda^{-2}$.

8. **Gaussian with parameters μ, σ^2 : $\mathcal{N}(\mu, \sigma^2)$**

   $f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\{-(x-\mu)^2/(2\sigma^2)\}$. Mean $= \mu$, variance $= \sigma^2$.

11 Appendix: Some Mathematical Facts

The following definitions and facts are used repeatedly in this course.

1. The following set notation is assumed to be familiar: $\emptyset, A \cap B, A \cup B, A \Delta B, A \setminus B, \bar{A} = A^c$.
2. $(A \cap B)^c = A^c \cup B^c$ and $(A \cup B)^c = A^c \cap B^c$.
3. $A \times B := \{(a,b) \mid a \in A, b \in B\}$.
4. $A^2 := A \times A$; $A^n := A^{n-1} \times A = \{(a_1, \ldots, a_n) \mid a_k \in A, k = 1, \ldots, n\}$.
5. A^* is the set of finite strings with elements in A. Thus $A^* := \cup_{n=0}^{\infty} A^n$ where A^0 is the set that contains one string of length 0.
6. For $a \ne 1$ and $n \ge 0$, one has $1 + a + \cdots + a^n = (1 - a^{n+1})/(1-a)$.
7. For $|a| < 1$, one has $1 + a + a^2 + \cdots = 1/(1-a)$.
8. $1 + 2a + 3a^2 + 4a^3 + \cdots = (d/da)[1 + a + a^2 + \cdots] = (d/da)(1-a)^{-1} = (1-a)^{-2}$.
9. For $0 \le m \le n$, one defines $\binom{n}{m} := \frac{n!}{m!(n-m)!}$.
10. For $n \ge 0$, one has $(a+b)^n = \sum_{m=0}^{n} \binom{n}{m} a^m b^{n-m}$.
11. $\ln(1+\varepsilon) \approx \varepsilon$ for $|\varepsilon| \ll 1$.
12. $\exp\{a\} = \sum_{n=0}^{\infty} \frac{a^n}{n!}$.
13. $\exp\{\varepsilon\} \approx 1 + \varepsilon$ for $|\varepsilon| \ll 1$.
14. $\exp\{a+b\} = \exp\{a\}\exp\{b\}$.
15. $\log_b(x) = \log_a(x) \log_b(a)$.
16. $1 + 2 + 3 + \cdots + n = n(n+1)/2$.

17. $\int_a^b f(x)dg(x) = f(b)g(b) - f(a)g(a) - \int_a^b g(x)df(x)$.

18. $\int_0^\infty \exp\{-ax\}dx = 1/a$ for $a > 0$.

19. $\int_{-\infty}^\infty \exp\{-\frac{x^2}{2}\}dx = \sqrt{2\pi}$.

20. $\int_{-\infty}^\infty x^2 \exp\{-\frac{x^2}{2}\}dx = \sqrt{2\pi}$.