1 Discrete Probability Essentials

We see probability everywhere. What’s the probability that if I toss a fair coin it will come up heads? Or the probability that if I roll a single die, I will roll a 3 or a 4? What is the probability that a particular runner will win a race against 9 other equally talented runners?[1]

All of these situations can be viewed as experiments that will produce one of a finite number of outcomes. We are interested in knowing how likely certain outcomes might be. In the die rolling case, we are interested in the likelihood of an event—a set of outcomes.

In order to reason accurately about probabilities, we model them mathematically. The basic model is pretty simple:

**Definition 1 (Finite Probability Space).** A finite probability space \((U, p)\) consists of a finite set \(U\) along with a function \(p : U \rightarrow \mathbb{R}\) such that

- For each \(u \in U\), \(p(u) \geq 0\), and
- \(\sum_{u \in U} p(u) = 1\).

That’s it! Just a set and a function capturing the notion of likelihood. In probability lingo, a subset \(A \subseteq U\) is called an event and an element \(u \in U\) is often called an outcome or an atomic or elementary event. The probability of any event is just the sum of the probabilities of it’s atomic events; that is, for any \(A \subseteq U\):

\[ p(A) = \sum_{u \in A} p(u). \]

Using basic set theory, of the kind you probably saw in middle school, we can establish some basic facts about probabilities.

**Proposition 1.** Let \((U, p)\) be a finite probability space and let \(A, B \subseteq U\). Then

1. If \(A \cap B = \emptyset\) then \(p(A \cup B) = p(A) + p(B)\)
2. \(p(A \cup B) = p(A \setminus B) + p(B \setminus A) + p(A \cap B)\) (prove this using previous item)
3. \(p(A \cup B) = p(A) + p(B) - p(A \cap B) \leq p(A) + p(B)\) (prove this using previous item)

**Example**

A pair of fair six-sided dice are rolled. What is the probability that the sum of the face up numbers is either 7 or 11?

[1] Answers: 1/2, 1/3, 1/10 (barring ties), respectively.
Solution. Our probability space is \((U, p)\), where \(U = \{(i, j) : 1 \leq i, j \leq 6\}\) and 
\[ p(u) = \frac{1}{36} \] for each \(u \in U\).

Let \(A\) be the event that a sum of 7 is rolled and \(B\) the event that a sum of 11 is rolled. Then
\[ A = \{(1, 6), (2, 5), (3, 4), (4, 3), (5, 2), (6, 1)\} \text{ and } B = \{(5, 6), (6, 5)\}, \]
so
\[ p(A \cup B) = p(A) + p(B) = \frac{6}{36} + \frac{2}{36} = \frac{2}{9} \] (since \(A\) and \(B\) are disjoint sets).

Example

A fair 10-sided die with sides numbered 1 through 10 is rolled. What is the probability that the number that comes face up is either prime or a multiple of three?

Solution. Our probability space is \((U, p)\), where \(U = \{1, 2, \ldots, 10\}\) and 
\[ p(u) = \frac{1}{10} \] for each \(u \in U\) (since the die is fair). Let \(A\) be the event that the number rolled is prime: 
\[ A = \{2, 3, 5, 7\} \text{ and } B = \{3, 6, 9\}. \]
We want 
\[ p(A \cup B) = p(A) + p(B) - p(A \cap B) = \frac{4}{10} + \frac{3}{10} - \frac{1}{10} = \frac{6}{10} = 0.6. \]

What is the probability that a randomly generated array of \(n\) integers has its values in increasing order (that is, is sorted)?

Solution. For any given set of \(n\) integers, they can be ordered in \(n!\) ways. Thus the probability that the array is sorted is \(\frac{1}{n!}\).

One more basic notion before we move on. If we roll a pair of dice, we believe that the dice do not influence one another: that is, the number that comes up on one of the dice has no bearing on the number that comes up on the other die. We formalize this notion as follows:

Example

**Definition 2.** Let \((U, p)\) be a finite probability space and let \(A, B \subseteq U\). \(A\) and \(B\) are independent events if 
\[ p(A \cap B) = p(A)p(B). \]

For example, in the dice rolling model, let \(A\) be the event that die 1 rolls an even number and \(B\) be the event that die 2 rolls a 5. Then 
\[ p(A) = \frac{18}{36} = \frac{1}{2}, p(B) = \frac{6}{36} = \frac{1}{6}, \text{ and } p(A \cap B) = \frac{3}{36} = \frac{1}{12}, \]
so 
\[ p(A \cap B) = p(A)p(B) \] and \(A\) and \(B\) are independent events. If \(B\) were the event that the sum of the numbers rolled was 10, then 
\[ p(B) = \frac{3}{36} = \frac{1}{12} \text{ and } p(A \cap B) = \frac{2}{36} = \frac{1}{18}, \]
so the events are not independent.

You might wonder why this is a reasonable definition of independence of events. Here’s the idea: If we believe that an event \(B\) is independent of another event \(A\), then it should be that 
\[ p(B) = \frac{p(A \cap B)}{p(A)}; \] in other words, restricting the set of all outcomes to only those in \(A\) should not change the probability that the outcome is in \(B\). This equation is

\[ \text{Do you agree that there are 36 possible outcomes? If not, pretend one die is red and the other is green. Does this change your opinion?} \]
equivalent to the equation in the definition above. This observation leads to the concept of conditional probability:

**Definition 3.** Let $(U, p)$ be a finite probability space and let $A, B \subseteq U$. The conditional probability of $B$ given $A$ is defined as $p(B|A) = p(B \cap A)/p(A)$.

Thus events $A$ and $B$ are independent if $p(B|A) = p(B)$: knowing $A$ occurred doesn’t change the probability of $B$ occurring.

## 2 Random Variables and Expected Value

Let’s play a game. Roll a fair 6-sided die. If any number other than 1 comes up, you win a dollar, but if a 1 comes up, you lose $4. If you were to roll the die $n$ times, how much might you expect to win (lose)? Well, with probability $1/6$, you will lose $4 and with probability $5/6$ you will win $1. Thus if you roll the die $n$ times, you might expect that on $n/6$ of the rolls you would lose $4, for a total loss of $4n/6, and on $5n/6$ of the rolls you would win $1 for total winnings of $5n/6. Thus, the expected result is -$4n/6 + 5n/6 = $n/6. Note that this kind of computation is most useful if you are going to roll several times: If $n = 1$, then the expected winnings of $1/6 isn’t very meaningful. For large values of $n$, though, this expected value is very informative.

Let’s briefly formalize the ideas of the previous paragraph. We’ll model the game with a probability space $U = \{w, l\}$ and probability measure $p$ given by $p(w) = 5/6$ and $p(l) = 1/6.$

How do we model the payout of the game? We define a function $X : U \rightarrow \mathbb{R}$, where $X(w) = 1$ and $X(l) = -4$. $X$ is called a random variable.

**Definition 4.** The expected value of a random variable $X$ defined on a probability space $(U, p)$ is given by

$$E[X] = \sum_{u \in U} X(u)p(u).$$

In our example we have $E[X] = X(w)p(w) + X(l)p(l) = 1 \cdot 5/6 + (-4) \cdot 1/6 = 1/6.$ Note that this gives the ”less useful” result of rolling the die once, but as we’ll see soon, we can get the more useful result just by multiplying $E[X]$ by the number of rolls. Here’s a very useful fact about random variables.

**Theorem 1 (Linearity of Expectation).** Let $X_1, \ldots, X_n$ be random variables defined on a probability space $(U, p)$. Then

$$E[X_1 + \ldots + X_n] = E[X_1] + \ldots E[X_n].$$

Back to our game: We can model rolling the die $n$ times by considering $n$ identical random variables $X_1, \ldots, X_n$ were each $X_i$ is the random variable $X$ defined previously.

---

1Note that we could have chosen $U = \{1, \ldots, 6\}$ and $p(u) = 1/6$ for each $u \in U$. The final result would have been the same.

2Oddly, perhaps, since it is neither random nor a variable....
Then the expected payout for \( n \) rolls of the die is
\[
E[X_1 + \ldots + X_n] = E[X_1] + \ldots E[X_n] = nE[X] = n/6
\]
dollars.

Example

Let’s apply all of this to an algorithm. Consider the problem of searching an unordered array \( a[] \) of \( n \) elements for a particular value \( x \) by starting with \( a[0] \) and checking consecutive entries until the element is found or until all entries have been checked. What is the expected number of elements of \( a[] \) that we will need to check if we know that \( a[] \) contains \( x \)?

Solution. We need to make some assumption about the likelihood that \( x \) is in some particular location in \( a[] \); absent any special information, we assume that it is equally likely that \( x \) is in any particular location. Let \( U = \{0, \ldots, n-1\} \) represent the indices of \( a[] \) and \( p(u) = 1/n \) for each \( u \in U \). Let \( X \) be the random variable representing the number of entries of \( a[] \) that are inspected by the algorithm: \( X(u) = u + 1 \), for each \( u \in U \). Then the expected number of entries of \( a[] \) inspected by the algorithm is
\[
E[X] = \sum_{u \in U} X(u) \cdot p(u) = \sum_{u=0}^{n-1} (u+1) \cdot 1/n = \frac{1 + \ldots + n}{n} = \frac{n+1}{2}
\]
\[\Box\]

Example

As another example, let’s try to figure out the expected running time of a binary search on a sorted \( n \) element array.

Solution. We will assume that the value is in the array, and that it is equally likely to be in any of the \( n \) positions. We can define \( U, X, \) and \( p \) as in the previous example and so \( E[X] = \sum_{u \in U} X(u) \cdot p(u) = 1/n \sum_{u \in U} X(u) \).

Unlike the previous example, coming up with an explicit formula for \( X(u) \) is a bit trickier. Instead, let’s just observe that there will be 1 value of \( u \) (namely, \((n-1)/2\)) for which \( X(u) = 1 \), 2 values for which \( X(u) = 2 \), 4 values for which \( X(u) = 3 \), etc.. If we assume that \( n = 2^k - 1 \) for some \( k \), then our sum becomes
\[
E[X] = 1/n \sum_{u \in U} X(u) = 1/n \sum_{i=1}^{k-1} 2^{i-1} = \frac{(k2^k - 2^k + 1)}{n}
\]
\[= ((n + 1) \log_2(n + 1) - n)/n \]
\[\approx \log_2(n + 1) - 1. \]
So, still \( O(\log_2 n) \) on average. All that work for nothing.... *Sigh* \[\Box\]
This last example suggests another way of expressing the expected value of a discrete random variable. If $X : U \to \mathbb{N}$ we can consider, for any $i \in \mathbb{N}$, the event $X_i$ consisting of all $u \in U$ such that $X(u) = i$. Then

$$E[X] = \sum_{i=0}^{\infty} ip(X_i)$$

Often the quantity $p(X_i)$ is written as $p[X = i]$ giving

$$E[X] = \sum_{i=0}^{\infty} ip(X = i) = \sum_{i=1}^{\infty} ip(X = i) \text{ (since the } i = 0 \text{ term drops out)}$$

Note that if $|U| = n$ is finite, then $X_i > 0$ for only finitely many values.

In the previous example, we could then have written

$$E[X] = \sum_{i=1}^{\log(n+1)} ip(X = i) = \sum_{i=1}^{\log(n+1)} ip(X = i) \cdot \frac{2^{i-1}}{n} = \ldots$$

Note that we stop the sum at $\log(n+1)$ since the probability that more than that many compares are needed equals 0!

**Example**

Let’s try one more analysis and to see whether we can figure out the expected running time of Bubble Sort. We’ll assume here that we are looking at the "optimized" version of Bubble Sort—the one that halts once all of the elements are sorted. Note that the running time of this algorithm should be roughly proportional to the number of swaps performed, so that is what we will focus on as the operations of interest.

**Solution.** Here our $U$ is the set of all permutations of the $n$ values held in the array—we will assume that all of the values are distinct. So $U$ contains $n!$ elements. We will let $X(u)$ count the number of swaps that Bubble Sort performs on input $u$. Note that $X()$ is just the number of pairs of elements that are "out of order" with respect to one another: that is, the number of pairs $\{i, j\}$ where $i < j$ but $a[i] > a[j]$. So our expected value equation is

$$E[X] = \sum_{u \in U} X(u) \cdot p(u) = \frac{1}{n!} \sum_{u \in U} X(u).$$

Now, just as in the previous problem, it is not easy to see a formula for $X(u)$, so we’ll try to identify a way to "re-group" the terms in our sum as we did in the previous problem. Consider some permutation $u$ of the elements in the array $a$. Let $u'$ be the reverse permutation—that is, the permutation with the elements of $a$ in the opposite

---

*Here the natural numbers include 0.*
order of $u$. Note that every pair of "out of order" elements of $u$ are "in order" in $u'$—
and vice versa. Thus, $X(u) + X(u') = n(n - 1)/2$; in other words, every pair of
elements is out of order in exactly one of $u$ and $u'$. Each permutation has a unique
reverse, so the $n!$ permutation in $U$ split into $n!/2$ pairs of the form $\{u, u'\}$. Now our
sum becomes

$$E[X] = \frac{1}{n!} \sum_{u \in U} X(u) = \frac{1}{n!} \sum_{\{u, u'\}} (X(u) + X(u'))$$  \hspace{1cm} (4)

$$= \frac{1}{n!} \sum_{\{u, u'\}} \frac{n(n - 1)}{2} = \frac{1}{n!} \frac{n(n - 1)}{2}$$  \hspace{1cm} (5)

$$= \frac{n(n - 1)}{4}.  \hspace{1cm} (6)$$

This tells us that Bubble Sort still takes time $O(n^2)$ on average—and, in fact, that it
should run in about half the time, on average, that it does in the worst case.