2 Statistical Principles

This lecture will serve two main goals. First we will introduce and the tool of random hash functions. Second we introduce a randomized/probabilistic view of algorithms and data analysis. This will include revisiting ideas about concentration of measure, and also probably approximate correct (PAC) error bounds.

We will study these properties through three phenomenon of random processes:

- **Birthday Paradox:** To measure the expected collision of random events.

  A random group of 23 people has about a 50% chance of having a pair with the same birthday.

- **Coupon Collectors:** To measure the expectation for seeing all possible outcomes of a discrete random variable.

  Consider a lottery where on each trial you receive one of n possible coupons at random. It takes in expectation about $n(0.577 + \ln n)$ trials to collect them all.

- **Central Limit Theorem:** To bound the leakage from the expected value of repeated random trials.

  Consider drawing ages (in 0 to 125) from an unknown distribution of people, how far is the sample average from the true average age.

From another perspective, these describe the effects of random variation. The first describes *collision* events, the second *covering* events, and the third *concentration* events.

2.0.1 Independent and Identically Distributed

What is data? A simplified view is a set $X = \{x_1, x_2, \ldots, x_n\}$, and in particular where each $x_i \sim D(\theta)$; that is each item is iid from some (probably unknown) distribution $D(\theta)$. The abbreviation *iid* means *independently and identically distributed*. In this context this means each x_i is (*identical*) drawn from the same distribution (in this case $D(\theta)$) and (*independent*) that the value of x_i is not affected by any other value x_j (for $i \neq j$).

This will not hold for all data sets. But its not uncommon for some aspect of the data to fit this model, or at least be close enough to this model. And for many problems without such an assumption, its impossible to try to recover the true underlying phenomenon that is the data mining objective. This is represented by the parameter(s) θ of the distribution $D(\theta)$.

In other settings, we will create an (randomized) algorithm that generates data which precisely follows this pattern (we can do this, since we design the algorithm!). Such an algorithm design is beneficial since we know a lot about iid data, this is the focus of most of the the classical part of the field of statistics!

In both cases, this is indicative of the very powerful ”big data” paradigm:

Create a complex/accurate estimate by combining many small and simple observations.

Model. For the setting of this lecture there is a common model of random elements drawn from a discrete universe. The universe has n possible objects; we represent this as $[n]$ and let $i \in [n]$ represent one element (indexed by i from $\{1, 2, 3, \ldots, n\}$) in this universe. The n objects may be IP addresses, days of the year, words in a dictionary, but for notational and implementation simplicity, we can always have each element (IP address, day, word) map to a distinct integer i where $0 < i \leq n$. Then we study the properties of drawing k items uniformly at random from $[n]$ with replacement.

2.1 (Random) Hash Functions

A key tool, perhaps *the* tool, in probabilistic algorithms is a hash function. There are two main types of hash functions. The second (which we define in Lecture 5) is locality-preserving. But often one wants a hash functions which is uncorrelated, like in a hash table data structure.

A *random hash function* $h \in \mathcal{H}$ maps from one set $\mathcal{A}$ to another $\mathcal{B}$ (usually $\mathcal{B} = [m]$) so that conditioned on the random choice $h \in \mathcal{H}$, the location $h(x) \in \mathcal{B}$ for any $x \in \mathcal{A}$ is equally likely. And for any two (or more strongly, but harder to achieve, *all*) $x \neq x' \in \mathcal{A}$ the $h(x)$ and $h(x')$ are *independent* (conditioned on the random choice of $h \in \mathcal{H}$). For a fixed $h \in \mathcal{H}$, the map $h(x)$ is *deterministic*, so no matter how many times we call h on argument x, it always returns the same result. The full independence requirement is often hard to achieve in theory, but in practice this difference is usually safe to ignore. For the purposes of this class, most built-in hash functions should work sufficiently well.

Assume we want to construct a hash function $h : \Sigma^k \to [m]$ where m is a power of two (so we can represent it as $\log_2 m$ bits) and Σ^k is a string of k of characters (lets say a string of numbers $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$). So $x \in \Sigma^k$ can also be interpreted as a k-digit number.

1. **SHA-1:** See `http://en.wikipedia.org/wiki/SHA-1`
   This secure hash function takes in a string of bits so $\Sigma = \{0, 1\}$ and k is the number of bits (but many front ends exists that can let Σ be all ASCII characters), and deterministically outputs a set of 160 bits, so $m = 2^{160}$.

   If one desired a smaller value of m, one can simply use any consistent subset of the bits. To create a family of hash functions $\mathcal{H}$ parameterized by some seed a (the *salt*), one can simply concatenate a to all inputs, to obtain a hash $h_a \in \mathcal{H}$. So $h_a(x) = \textsf{SHA-1}(\textsf{concat}(a, x))$.

   Functionality for this (or other similar hash functions) should be built in or available as standard packages for many programming languages. These built in functions often can take in strings and other various inputs which are then internally converted to bits.

2. **Multiplicative Hashing:** $h_a(x) = \lfloor m \cdot \textsf{frac}(x * a) \rfloor$
   a is a real number (it should be large with binary representation a good mix of 0s and 1s), $\textsf{frac}(\cdot)$ takes the fractional part of a number, e.g. $\textsf{frac}(15.234) = 0.234$, and $\lfloor \cdot \rfloor$ takes the integer part of a number, rounding down so $\lfloor 15.234 \rfloor = 15$. Can sometimes be more efficiently implemented as $(xa/2^q) \mod m$ where q is essentially replacing the $\textsf{frac}(\cdot)$ operation and determining the number of bits precision.

   If you want something simple and fast to implement yourself, this is a fine choice. In this case the randomness is the choice of a. Once that is chosen (randomly), then the function $h_a(\cdot)$ is deterministic.

3. **Modular Hashing:** $h(x) = x \mod m$
   *(This is **not recommended**, but is a common first approach. It is listed here to advise against it.)*

   This roughly evenly distributes a number x to $[m]$, but numbers that are both the same $\mod m$ *always* hash to the same location. One can alleviate this problem by using a random large prime $m' < m$. This will leave bin $\{m' + 1, m' + 1, \ldots, m - 1, m\}$ always empty, but has less regularity.

2.2 Birthday Paradox

First, let us consider the famous situation of birthdays. Lets make formal the setting. Consider a room of k people, chosen at random from the population, and assume each person is equally likely to have any birthday (excluding February 29th), so there are $n = 365$ possible birthdays.

The probability that any two (i.e. $k = 2$) people (Alice and Bob) have the same birthday is $1/n = 1/365 \approx 0.003$. The birthday of Alice could be anything, but once it is known by Alice, then Bob has probability $1/365$ of matching it.

To measure that at least one pair of people have the same birthday, it is easier to measure the probability that no pair is the same. For $k = 2$ the answer is $1 - 1/n$ and for $n = 365$ that is about 0.997.

For a general number k (say $k = 23$) there are $\binom{k}{2} = k \cdot (k-1)/2$ (read as k *choose* 2) pairs. For $k = 23$, then $\binom{23}{2} = 253$. Note that $\binom{k}{2} = \Theta(k^2)$.

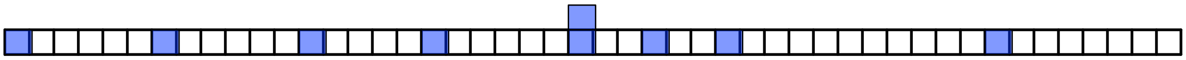
We need for each of these events that the birthdays do not match. Assuming independence we have

$$(1 - 1/n)^{\binom{k}{2}} \quad \text{or} \quad 0.997^{253} = 0.467.$$

And the probability there is a match is thus 1 minus this number

$$1 - (1 - 1/n)^{\binom{k}{2}} \quad \text{or} \quad 1 - 0.997^{253} = 0.532,$$

just over 50%.



What are the problems with this?

- First, the birthdays may not be independently distributed. More people are born in spring. There may be non-negligible occurrence of twins.

  Sometimes this is really a problem, but often it is negligible. Other times this analysis will describe an algorithm we create, and we can control independence.

- Second, what happens when $k = n+1$, then we should always have some pair with the same birthday. But for $k = 366$ and $n = 365$ then

  $$1 - (1 - 1/n)^{\binom{k}{2}} = 1 - (364/365)^{\binom{366}{2}} = 1 - (0.997)^{66795} = 1 - 7 \times 10^{-88} < 1.$$

  Yes, it is very small, but it is less than 1, and hence must be wrong.

  Really, the probability should be

  $$1 - \left(\frac{n-1}{n}\right) \cdot \left(\frac{n-2}{n}\right) \cdot \left(\frac{n-3}{n}\right) \cdot \ldots = 1 - \prod_{i=1}^{k-1} \left(\frac{n-i}{n}\right).$$

Inductively, in the first round (the second person ($i = 2$) there is a $(n-1)/n$ chance of having no collision. If this is true, we can go to the next round, where there are then two distinct items seen, and so the third person has $(n-2)/n$ chance of having a distinct birthday. In general, inductively, after the ith round, there is an $(n-i)/n$ chance of no collision (if there were no collisions already), since there are i distinct events already witnessed.

As a simple sanity check, in the $(n+1)$th term $(n-n)/(n) = 0/n = 0$; thus the probability of some collision of birthdays is $1 - 0 = 1$.

Take away message.

- There are collisions in random data!
- More precisely, if you have n equi-probability random events, then expect after about $k = \sqrt{2n}$ events to get a collision. Note $\sqrt{2 \cdot 365} \approx 27$, a bit more than 23.

  Note that $(1 + \frac{\alpha}{t})^t \approx e^\alpha$ for large enough t. So setting $k = \sqrt{2n}$ then

  $$1 - (1 - 1/n)^{\binom{k}{2}} \approx 1 - (1-1/n)^n \approx 1 - e^{-1} \approx .63$$

  This is not exactly $1/2$, and we used a bunch of $\approx$ tricks, but it shows *roughly* what happens.
- This is fairly accurate, but has noticeable variance. Note for $n = 365$ and $k = 18$ then

  $$1 - (1 - 1/n)^{\binom{k}{2}} = 1 - (364/365)^{153} \approx .34$$

  and when $k = 28$ then

  $$1 - (1 - 1/n)^{\binom{k}{2}} = 1 - (364/365)^{378} \approx .64.$$

  This means that if you keep adding (random) people to the room, the first matching of birthdays happens 30% (= 64% − 34%) of the time between the 18th and 28th person. When $k = 50$ people are in the room, then

  $$1 - (1 - 1/n)^{\binom{k}{2}} = 1 - (364/365)^{1225} \approx .965,$$

  and so only about 3.5% percent of the time are there no pair with the same birthday.

2.3 Coupon Collectors

Lets now formalize the famous coupon lottery. There are n types of coupons, and we participate in a series of independent trials, and on each trial we have equal probability $(1/n)$ of getting each coupon. *We want to collect all toys available in a McDonald's Happy Meal.* How many trials (k) should we expect to partake in before we collect all coupons?

Let r_i be the expected number of trials we need to take before receiving exactly i *distinct* coupons. Let $r_0 = 0$, and set $t_i = r_i - r_{i-1}$ to measure the expected number of trials between getting $i-1$ distinct coupons and i distinct coupons.

Clearly, $r_1 = t_1 = 1$, and it has no variance. Our first trials always yields a new coupon.

Then the expected number of trials to get all coupons is $T = \sum_{i=1}^n t_i$.

To measure t_i we will define p_i as the probability that we get a new coupon after already having $i-1$ distinct coupons. Thus $t_i = 1/p_i$. And $p_i = (n-i+1)/n$.

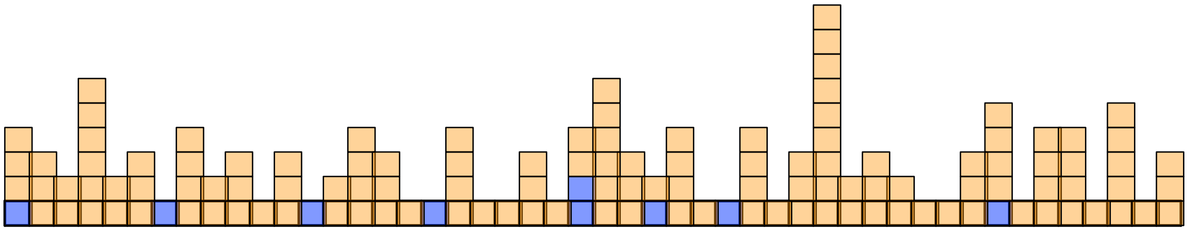
We are now ready for some algebra:

$$T = \sum_{i=1}^n t_i = \sum_{i=1}^n \frac{n}{n-i+1} = n \sum_{i=1}^n \frac{1}{i}.$$

Now we just need to bound the quantity $\sum_{i=1}^{n}(1/i)$. This is known at the nth *Harmonic Number* H_n. It is known that $H_n = \gamma + \ln n + o(1/n)$ where $\ln(\cdot)$ is the natural log (that is $\ln e = 1$) and $\gamma \approx 0.577$ is the *Euler-Masheroni constant*. Thus we need, in expectation,

$$k = T = nH_n \approx n(\gamma + \ln n)$$

trials to obtain all distinct coupons.



Extensions.

- What if some coupons are more likely than others. *McDonalds offers three toys: Alvin, Simon, and Theodore, and for every 10 toys, there are 6 Alvins, 3 Simons, and 1 Theodore.* How many trials do we expect before we collect them all?

  In this case, there are $n = 3$ probabilities $\{p_1 = 6/10, p_2 = 3/10, p_3 = 1/10\}$ so that $\sum_{i=1}^{n} p_i = 1$.

  The analysis and tight bounds here is a bit more complicated, but the key insight is that it is dominated by the smallest probability event. Let $p^* = \min_i p_i$. Then we need about

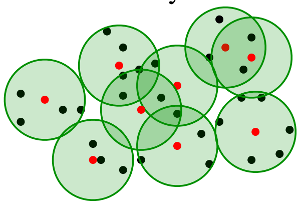
  $$k \approx \left(\frac{1}{p^*}\right)(\gamma + \ln n)$$

  random trials to obtain all coupons.

- These properties can be generalized to a family of events from a continuous domain. Here there can be events with arbitrarily small probability of occurring, and so the number of trials we need to get all events becomes arbitrarily large (following the above non-uniform analysis). So typically we set some probability $\varepsilon \in [0,1]$. (Typically we consider ε as something like $\{0.01, 0.001\}$ so $1/\varepsilon$ something like $\{100, 1000\}$. Now we want to consider any set of events with combined probability greater than ε. (We can't consider all such subsets, but we can restrict to all, say, contiguous sets – intervals if the events have a natural ordering). Then we need

  $$k \approx \frac{1}{\varepsilon}\log\frac{1}{\varepsilon}$$

  random trials to have at least one random trial in any subset with probability at least ε. Such a set is called an *ε-net*.



Take away message.

- It takes about $n \ln n$ trials to get all items at random from a set of size n, not n. That is we need an extra about $\ln n$ factor to probabilistically guarantee we hit all events.

- When probabilities are not equal, it is the smallest probability item that dominates everything!

- To hit all (nicely shaped) regions of size εn we need about $(1/\varepsilon)\log(1/\varepsilon)$ samples, even if they can be covered by $1/\varepsilon$ items.

2.4 Probably Approximately Correct (PAC)

The above discussions are starting to hint at a *probability approximate correct* (PAC) bound. There are random events that produce an estimate $\hat{X}$ (which is a random variable) that is probably close to its expected value $\mu = \mathbf{E}[\hat{X}]$.

The tricky part of about describing this is that we don't want to say $\hat{X}$ is always close to μ. Because it is a random process, sometimes things go (horribly!) wrong. So we want to provide a term $\delta \in (0,1)$ which is a probability of failure. Sometimes (with say $\delta = 0.01$ probability of failure, or 1% of the time) $\bar{X}$ is *not* close to μ.

Moreover, we also don't want to talk about the probability that $\bar{X} = \mu$. Since this may occur with very small probability (we need exactly the right number of happy means, maybe 1000 to collect all 100 or so coupons). Rather we would like to allow some error threshold, using a parameter $\varepsilon > 0$.

So we need *two* parameters to describe the error δ (probably) and ε (approximate). Ultimately most bounds look of the form:

$$\mathbf{Pr}[|\hat{X} - \mu| \geq \varepsilon] \leq \delta.$$

That is, the probability that $\hat{X}$ (which is some random variable, often a sum of iid random variables) is further than ε to its expected value μ, is at most δ. Or equivalently (perhaps more optimistically)

$$\mathbf{Pr}[|\hat{X} - \mu| < \varepsilon] > 1 - \delta.$$

I like to think of ε as the *error tolerance* and δ as the *probability of failure* i.e., that we exceed the error tolerance. However, often these bounds will allow us to write the required sample size n in terms of ε and δ. This allows us to trade these two terms off for any fixed known n; we can allow less error tolerance if we are willing to allow more probability of failure, and vice-versa.

2.5 Central Limit Theorem and Chernoff-Hoeffding Bound

When dealing with modern big data sets, a very common theme is reducing the set through a random process. These generally work by making "many simple estimates" of the full data set, and then judging them as a whole. Perhaps magically, these "many simple estimates" can provide a very accurate and small representation of the large data set. In statistics, this phenomenon is usually referred to as the Central Limit Theorem. The key tool in showing how many of these simple estimates are needed for a fixed accuracy trade-off, and formulating the Central Limit Theorem as a PAC bound, is the *Chernoff-Hoeffding* inequality.

We consider a specific form of the Chernoff-Hoeffding inequality. It is not the strongest form of the bound, but is for many applications asymptotically equivalent, and it also fairly straight-forward to use. It is more similar to the form of Azuma's inequality which deals with Martingales that have more complicated dependence structure.

Theorem 2.5.1. *Consider a set of r independent identically distributed (iid) random variables $\{X_1, \ldots, X_r\}$ such that $-\Delta \leq X_i \leq \Delta$ for each $i \in [r]$. Let $A = \frac{1}{r}\sum_{i=1}^r X_i$ (average of X_is). Then for any $\alpha > 0$*

$$\textbf{\textit{Pr}}[|A - \textbf{\textit{E}}[A]| > \alpha] \leq 2\exp\left(\frac{-r\alpha^2}{2\Delta^2}\right).$$

This requires us to know two things about the random variables. First, we should know the expected value of $\mathbf{E}[X_i]$ for all X_i; since they are iid, its the same for all of them. Second, we need to know an upper bound Δ on the range of these random variables. For instance, if its someones age, we can probably bound $\Delta = 125$ since no one is older than 125 years old, and all ages are positive so $0 > -\Delta = -125$. This tells us how far we can reasonably expect our sample estimate A of the average age in a set to be from the true average. Thus, we may not even need to know $\mathbf{E}[A]$, the true expected age, but just want to know how likely it is that we are very far from it.

For instance, if we find $A = 30$, and $r = 1000$, then the probability that $\mathbf{E}[A] \in [20, 40]$ (so $\alpha = 10$) is at most $2\exp(-1000 \cdot 10^2/(2 \cdot 125^2)) \approx 0.08$. So (at most) about 8% of the time we would have an estimate *not* within 10 years. (Note, this sounds very pessimistic, since it could contain the high variance case where p fraction of the the person is 125 and $(1-p)$ fraction they are -125. Its important to try to get a tight estimate on the actual range of deviation.

More general form. This is a slightly more general form that I often find very useful as well.

Theorem 2.5.2. *Consider a set of r* independent *random variables $\{X_1, \ldots, X_r\}$. If we know $a_i \leq X_i \leq b_i$, then let $\Delta_i = b_i - a_i$. Let $M = \sum_{i=1}^r X_i$. Then for any $\alpha > 0$*

$$\textbf{\textit{Pr}}[|M - \textbf{\textit{E}}[M]| > \alpha] \leq 2\exp\left(\frac{-2\alpha^2}{\sum_{i=1}^r \Delta_i^2}\right).$$

For the same problem above, since we can use now each $\Delta_i = 125$ reveals a probability of at most 0.0000055 that the estimate is not within 10 years of the true value.

Hashing. These bounds map back to the hashing problem, to understand the case when all of the buckets have roughly the same number of items that fall in them. This requires roughly $k = (1/\varepsilon^2)\log(1/\delta)$ to be hashed so that each bucket is within $\varepsilon k/n$ of all other buckets, with probability at least $1 - \delta$.