Chapter 1

Streaming Algorithms and Concentration Inequalities

In these lectures we introduce the streaming model of computation and give a couple algorithms for estimating frequency moments. We review basic concentration inequalities for the analysis of randomized algorithms. We discuss the "median-of-means" technique for error and success amplification, and analyze it using the Chernoff bound. We give two applications of the Chernoff bound, to the Johnson-Lindenstrauss lemma for dimension reduction and to the balanced allocation problem.

1.1 The streaming model

The streaming model is as follows. The input is a "stream" of values $\sigma = (s_1, s_2, \ldots)$, where $s_j \in \{1, 2, \ldots, n\}$ for all j. The algorithm sees the values one by one. When a new value, s_j, arrives, it can perform an update. Then the value disappears, and the next value, s_{j+1}, arrives. The goal is that once all values have been seen, the algorithm should return a quantity of interest — for example, the number of distinct elements in the stream.

We will usually assume that the length m of the stream, and the range $\{1, \ldots, n\}$ of the stream elements, are known in advance. The main complexity parameter is the memory, S, that the algorithm requires. Sometimes the time T to perform the update for each new stream element is also measured.

Streaming algorithms are relevant in scenario where there is a massive flux of data coming through, so it is impossible to store everything in memory. But we would still like to compute some property of the data. An example is internet traffic, where we can't simply log every data packet that goes through, but would still like to maintain some rough statistics that would let us e.g. detect suspicious activity. Or data collected from large Physics experiments such as those at CERN, where we want to detect special events without having the ability to store all the data generated by the measurement apparatus. Or video feeds from surveillance cameras, where we want to spot suspicious individuals but can't store the whole stream and must process information in real time.

1.1.1 Estimating the number of distinct elements

As a warm-up, let's consider the problem of computing the number of distinct elements in a stream. This is called the 0-th moment of the stream, F_0. In general we denote by

$$f_i = \left|\{1 \le j \le m : s_j = i\}\right|$$

the "frequency" of a value $i \in \{1, \ldots, n\}$. Then $F_0 = \sum_{i=1}^{m} (f_i)^0$. Another example is $F_1 = \sum_i f_i = m$, which just counts the length of the stream.

Let's first try to compute F_0 using a deterministic procedure. How much space do you need? The most naïve algorithm stores a copy of each new element we see. In the worst case all elements are distinct, so this will require space $S = \min\{n, m\} \log n$. Can we do better?

Suppose you had a deterministic algorithm that solves the problem in space S. For $x \in \{0,1\}^n$ consider the stream $((1, x_1), \ldots, (n, x_n))$. This stream has n distinct elements. Suppose we execute the algorithm; at the end of the stream the state of the algorithm's memory is some arbitrary string $m(x) \in \{0,1\}^S$. Now let's initialize the algorithm's memory to $m(x)$ (for some x) and give it a single element $(i, 0)$. If the number of distinct elements reported by the algorithm increases by 1 then we know that $x_i = 1$, whereas if it doesn't change we know that $x_i = 0$. In this way it is possible to completely recover x from $m(x)$, therefore $m(x)$ must take at least 2^n possible values: $S \ge n$.

This argument only applies to deterministic algorithms that return an exact count for the number of distinct elements. What about an algorithm that *approximates* F_0, say up to 10% error? In your homework you will see that even in this case one needs space $\Omega(n)$.

To save on space we'll have to abandon determinism, and look for a *randomized* algorithm. This means the algorithm can toss coins, and we want the answer to be accurate with high probability over the outcomes of the coin tosses, for any input stream σ. So let f be a random function $h : \{1, \ldots, n\} \to \{1, \ldots, n\}$, which represents the outcomes of n n-valued coin tosses, and consider the following procedure:

(1) **Initialization:** Set $z = 0$.

(2) **Process s_j:** Let ℓ be the largest power of 2 that divides $h(s_j)$. If $\ell > z$ then $z \leftarrow \ell$.

(3) **Output:** Return $2^{z+\frac{1}{2}}$.

Why does this work? Suppose the stream has at least d distinct elements. Then there is a good chance that one of the d possible values $h(s_j)$ is divisible by $\log d$ (note that ℓ is simply the number of trailing zeros in the binary representation of $h(s_j)$). On the other hand, if the stream has no more than d distinct elements then it is pretty unlikely that there is an item that has much more than $\log d$ zeros in a row. This heuristic argument shows that we expect z to be a reasonable approximation to d, up to a constant multiplicative factor.

What is the space requirement of the procedure? Storing z only requires about $\log n$ bits, so this is very efficient. Note though that we are allowing the algorithm to store h "for free": we did not count the space required to record the coin tosses. If we count those as well, then we need an additional $n \log n$ bits — no better than the perfect deterministic algorithm

we saw earlier. This could be a serious issue, but we'll set it aside for now. Later we'll see how it is possible to do much better by using a partially random h — an h that is chosen randomly from a much smaller family of functions than all functions.

How do we analyze this? What is the probability that the algorithm returns an estimate that is correct within relative error 10%? What if we want error 1%, what is the overhead in terms of space requirements? Before we answer these questions, let's work out another example.

1.1.2 Estimating the second frequency moment

The second frequency moment is $F_2 = \sum_i f_i^2$. The second moment gives a measure of "discrepancy" within the stream: the higher F_2 the less "uneven" the frequencies are. Consider the following algorithm, where $h : \{1, \ldots, n\} \to \{\pm 1\}$ is again a random function:

(1) **Initialization:** Set $c = 0$.

(2) **Process s_j:** Add $h(s_j)$ to c.

(3) **Output:** Return c^2.

Note the space requirement for this algorithm is quite small (if we again store the function h for free): at each step c either increases or decreases by 1, so after m elements it ranges in $\{-m, \ldots, 0, \ldots, m\}$, requiring only $\log(2m + 1)$ bits to store.

How do we analyze the algorithm. Let $Z = c^2$ be a random variable containing the output of the algorithm. At a minimum we would like that the algorithm does well on average, meaning we want to compute $\mathbf{E}[Z]$ and compare it to F_2. To do this, for each $j \in \{1, \ldots, n\}$ introduce a random variable $Y_j = h(j) \in \{\pm 1\}$, so that $Z = (f_1 Y_1 + \cdots + f_n Y_n)^2$. We can then compute the expectation of Z as follows:

$$\begin{aligned}
\mathbf{E}[Z] &= \mathbf{E}[(f_1 Y_1 + \cdots + f_n Y_n)^2] \\
&= \sum_j f_j^2 \, \mathbf{E}[Y_j^2] + \sum_{i \neq j} f_i f_j \, \mathbf{E}[Y_i Y_j] \\
&= \sum_j f_j^2 + \sum_{i \neq j} f_i f_j \, \mathbf{E}[Y_i] \, \mathbf{E}[Y_j] \\
&= F_2,
\end{aligned}$$

where we used the fact that the Y_j are independent and $\mathbf{E}[Y_j] = 0$ for every j. This means the algorithm does return an unbiased estimator. How good is it? Could it be that $Z = 100F_2$ for half the time? Well, that's impossible: since always $Z \geq 0$, you can see that $Z \geq 100F_2$ can happen at most 1% of the time; otherwise even by setting it to 0 the remaining time it wouldn't have the right expectation. This is called Markov's inequality.

1.2 Concentration inequalities

1.2.1 Markov's Inequality

Suppose given a random variable Z such that the only assumption is $Z \geq 0$. Then Markov's inequality already lets us say something nontrivial:

Theorem 1.1 (Markov)**.** *For a random variable $Z \geq 0$ and any $k > 0$,*

$$\mathbf{Pr}(Z \geq k) \leq \frac{\mathbf{E}[Z]}{k}.$$

By choosing $k = \alpha \mathbf{E}[Z]$ for $\alpha \geq 1$ we can reformulate the bound as $\mathbf{Pr}(Z \geq \alpha \mathbf{E}[Z]) \leq \alpha^{-1}$

Proof. $\mathbf{E}[Z] \geq k \mathbf{Pr}(Z \geq k) + 0 \mathbf{Pr}(Z < k)$. □

We can also do a "proof by picture". Consider the function $g(Z) = \frac{Z}{k}$ and let $f(Z)$ be an indicator function which is 1 if $Z \geq k$ and 0 otherwise. Then $\mathbf{Pr}(Z \geq k) = \mathbf{E}[f(Z)] \leq \mathbf{E}[g(Z)] = \frac{\mathbf{E}[Z]}{k}$.

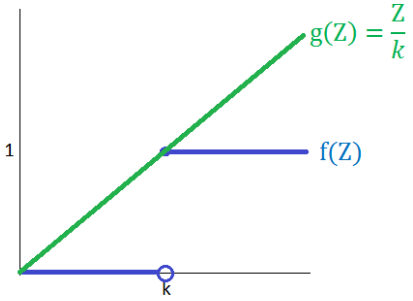

Figure 1.1: $g(Z) \geq f(Z)$ everywhere

Exercise 1. For every $k \geq 1$, give an example of a random variable Z such that Markov's inequality is tight for this choice of k.

Applied to our problem of estimating F_2, Markov's inequality gives $\mathbf{Pr}[Z \geq \alpha F_2] \leq \frac{1}{\alpha}$, for any $\alpha > 0$. So for instance the chance that we overshoot by a factor 4 is at most $1/4$. Can we do better? The exercise shows that, if we don't assume any more information on Z, we can't. To show that Z remains close to its expectation more frequently than the worst-case behavior, we have to compute the variance.

1.2.2 Chebyshev's Inequality

The variance of a random variable Z (not necessarily non-negative) is defined as

$$\mathbf{Var}[Z] = \mathbf{E}[(Z-\mu)^2], \qquad \text{where } \mu = \mathbf{E}[Z].$$

Theorem 1.2 (Chebyshev)**.** *For a random variable with mean* $\mathbf{E}[Z] = \mu$ *and any* $t \geq 0$*,*

$$\mathbf{Pr}\left(|Z-\mu| \geq t\right) \leq \frac{\mathbf{E}[(Z-\mu)^2]}{t^2}.$$

Proof. Note that the events $|Z-\mu| \geq t$ and $(Z-\mu)^2 \geq t^2$ are equivalent. Since the random variable $|Z-\mu|$ is non-negative, we can apply Markov's inequality:

$$\begin{aligned}\mathbf{Pr}\left(|Z-\mu| \geq t\right) &= \mathbf{Pr}\left((Z-\mu)^2 \geq t^2\right) \\ &\leq \frac{\mathbf{E}[(Z-\mu)^2]}{t^2}.\end{aligned}$$

□

Here again we can give a proof by picture with $g(Z) = \frac{(Z-\mu)^2}{t^2}$ and $f(Z)$ an indicator function which is 1 if $|Z-\mu| \geq t$ and 0 otherwise:

$$\begin{aligned}\mathbf{Pr}\left(|Z-\mu| \geq t\right) &= \mathbf{E}[f(Z)] \\ &\leq \mathbf{E}[g(Z)] \\ &= \mathbf{E}\left[\frac{(Z-\mu)^2}{t^2}\right] \\ &= \frac{\mathbf{E}[(Z-\mu)^2]}{t^2}.\end{aligned}$$

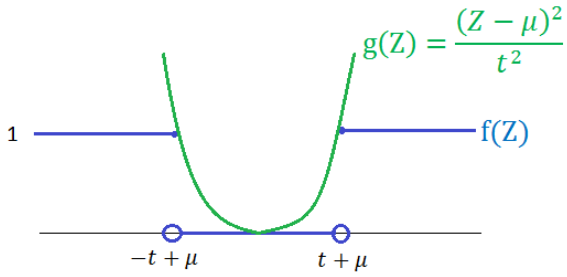

Figure 1.2: $g(Z) \geq f(Z)$ everywhere

From these pictorial proofs one can see that Chebyshev's inequality is just Markov's inequality with a different definition of $g(Z)$. Any function g which is larger than the indicator function will provide a different bound, and depending on the context it might be possible to get a better bound than Markov's or Chebyshev's.

Exercise 2. For every expectation $\mu \geq 0$, $t \geq 0$ and variance $0 \leq \sigma^2 \leq t^2/2$, give a random variable which has that expectation and variance and satisfies $\mathbf{Pr}\left(|Z - \mu| \geq t\right) = t^{-2}\,\mathbf{E}[(Z-\mu)^2]$.

1.3 Variance analysis

1.3.1 The second moment

To estimate the variance, we compute

$$\mathbf{E}[Z^2] = \mathbf{E}[(f_1 Y_1 + \cdots + f_n Y_n)^4] = \sum_{i,j,k,\ell} f_i f_j f_k f_\ell\, \mathbf{E}[Y_i Y_j Y_k Y_\ell].$$

Using independence, the only nonzero terms in the sum above are those where either $i = j = k = \ell$, or when two indices occur twice each. So

$$\mathbf{E}[Z^2] = \sum f_j^4\, \mathbf{E}[Y_j^4] + 3\sum_{i \neq j} f_i^2 f_j^2\, \mathbf{E}[Y_i^2 Y_j^2] = F_4 + 3(F_2^2 - F_4).$$

Thus

$$\mathbf{Var}[Z] = F_4 + 3(F_2^2 - F_4) - F_2^2 \leq 2F_2^2.$$

Applying Chebyshev's inequality, for any $\varepsilon > 0$,

$$\mathbf{Pr}(|Z - F_2| \geq \varepsilon F_2) \leq \frac{2F_2^2}{\varepsilon^2 F_2^2} = \frac{2}{\varepsilon^2}.$$

If we take $\varepsilon = 99$, then we see $\mathbf{Pr}(Z \geq 100F_2) \leq 2/99^2$, a much better bound than we got by Markov's inequality! There are still reasons to be dissatisfied, however. In particular, for the bound to be non-trivial we need to take $\varepsilon \geq \sqrt{2}$: what if we want a better estimate, say to within 1%? Later we'll see a general method to improve the quality of this approximation by maintaining a small number of independent counters. The result will be the following theorem:

Theorem 1.3. *For any $\varepsilon, \delta > 0$ there is a streaming algorithm using space $O(\frac{1}{\varepsilon^2}\log(1/\delta)\log m)$ that returns an estimate for F_2 which is accurate up to a multiplicative factor $(1 \pm \varepsilon)$ with probability at least $1 - \delta$.*

1.3.2 The 0-th moment

Recall that $F_0 = \sum f_i^0$ is the number of distinct elements in the stream. Let's analyze the success probability of the algorithm we saw earlier. For each $j \in \{1, \ldots, n\}$ and $r \geq 0$ let $X_{r,j}$ be the indicator random variable for the event that $h(j)$ is divisible by 2^r, and $Y_r = \sum_{j : f_j > 0} X_{r,j}$. Let t be the final value of z at the end of the algorithm. Note that $Y_r = 0$

means no element had r or more zeros, thus $t \leq r-1$. Using that, since h is a random function, for every j the value $h(j)$ is uniformly distributed in $\{1,\ldots,n\}$,

$$\mathbf{E}[X_{r,j}] = \mathbf{Pr}\left(2^r \text{ divides } h(j)\right) = \frac{1}{2^r}.$$

By linearity of expectation, $\mathbf{E}[Y_r] = F_0/2^r$, and using independence

$$\mathbf{Var}[Y_r] = \sum_{j:f_j>0} \mathbf{Var}[X_{r,j}] \leq \sum_{j:f_j>0} \mathbf{E}[X_{r,j}^2] = \sum_{j:f_j>0} \mathbf{E}[X_{r,j}] = \frac{F_0}{2^r}.$$

Using first Markov and then Chebyshev,

$$\mathbf{Pr}\left(Y_r > 0\right) = \mathbf{Pr}(Y_r \geq 1) \leq \frac{\mathbf{E}[Y_r]}{1} = \frac{F_0}{2^r},$$
$$\mathbf{Pr}(Y_r = 0) \leq \mathbf{Pr}(|Y_r - \mathbf{E}[Y_r]| \geq \frac{F_0}{2^r}) \leq \frac{\mathbf{Var}[Y_r]}{(F_0/2^r)^2} \leq \frac{2^r}{F_0}.$$

Let $\hat{F} = 2^{z+\frac{1}{2}}$ be the output of the algorithm, and a and b the smallest and largest integer such that $2^{a+\frac{1}{2}} \geq 3F_0$ and $2^{b+\frac{1}{2}} \leq F_0/3$ respectively. Then

$$\mathbf{Pr}(\hat{F} \geq 3F_0) = \mathbf{Pr}(z \geq a) = \mathbf{Pr}(Y_a > 0) \leq \frac{F_0}{2^a} \leq \frac{\sqrt{2}}{3},$$
$$\mathbf{Pr}(\hat{F} \leq F_0/3) = \mathbf{Pr}(z \leq b) = \mathbf{Pr}(Y_{b+1} = 0) \leq \frac{2^{b+1}}{F_0} \leq \frac{\sqrt{2}}{3}.$$

So we have obtained a factor 3-approximation to F_0 that is correct with probability $1-2\sqrt{2}/3$, by the union bound. But the success probability is pretty low, about 6%! However, just as for F_2 it is possible to boost this very quickly: for any ε, δ we can reduce the error to $\pm\varepsilon F_0$ and boost the success probability to $1-\delta$ using only $O(\varepsilon^{-2}\log(1/\delta))$ independent repetitions.

1.4 Derandomization

Before we proceed with the advanced analysis of our algorithms, let's deal with the problem of storing the random function h used by both algorithms. How do we deal with this? Observe that our analysis used the fact that h is a random function in a rather weak way: specifically, when computing the expectation and variance of the random variable $Z = c^2$ describing the outcome of the algorithm we used conditions such as $\mathbf{E}[Y_iY_jY_kY_\ell] = \mathbf{E}[Y_i]\,\mathbf{E}[Y_j]\,\mathbf{E}[Y_k]\,\mathbf{E}[Y_\ell]$ for distinct values i,j,k,ℓ, where $Y_j = h(j)$ is the random variable that describes the output of the function at a particular point. As it turns out, this requirement is a much weaker requirement than full independence, called 4-*wise independence*. In particular, it is possible to sample "4-wise independent" functions using much fewer random bits than a uniformly random function. This is the idea behind *derandomization*: to try to save on the number of random coins needed while keeping the function h "random enough" that the analysis carries over.

1.4.1 k-wise independent random variables and hash functions

Definition 1.4. A family of random variables $(X_1, \ldots, X_N)$ is called k-wise independent if for every k-tuple $(i_1, \ldots, i_k) \in \{1, \ldots, N\}^k$ the random variables $(X_{i_1}, \ldots, X_{i_k})$ are independent.

To see the difference between 2-wise (also called pairwise) independence and full independence, consider for example a triple of random variables (X_1, X_2, X_3) that is uniformly distributed over $\{(1, 1, -1), (1, -1, 1), (-1, 1, 1), (-1, -1, -1)\}$. Then (X_1, X_2, X_3) are certainly not independent: the product is always -1. But you can check that they are pairwise independent. Note how this lets us save on the randomness: to generate a sample (X_1, X_2, X_3) we only need 2 random bits, instead of 3 for fully independent random variables.

As an immediate consequence of the definition, you can see that if the Y_i are four-wise independent then the equality $\mathbf{E}[Y_i Y_j Y_k Y_\ell] = \mathbf{E}[Y_i]\,\mathbf{E}[Y_j]\,\mathbf{E}[Y_k]\,\mathbf{E}[Y_\ell]$ always holds, which is all that was needed for the analysis of our F_2 algorithm: we only need the values produced by h to be 4-wise independent, not fully independent. Here is a reformulation of this requirement:

Definition 1.5. A family $\mathcal{H}$ of functions $h : A \mapsto B$ is called k-wise independent if for any distinct points $x_1, ..., x_k \in A$ and $i_1, ..., i_k \in B$,

$$\Pr_{h \in \mathcal{H}} \big(h(x_1) = i_1, \ldots, h(x_k) = i_k\big) = \frac{1}{|B|^k}.$$

A 1-wise independent family of hash functions is just a family such that any element x in the domain is mapped to a random element in the range, when the function is chosen at random. In general, the requirement of $\mathcal{H}$ being k-wise independent is the same as requiring that the random variables $X_i = h(x_i)$, sampled by evaluating a random $h \leftarrow \mathcal{H}$ at a fixed point x_i, are k-wise independent random variables.

Example. For $A = B = \{0, 1, ..., p-1\}$ where p is a prime number, consider the following family of functions:

$$\mathcal{H}_2 = \big\{ f_{a,b} : x \mapsto ax + b \mod p,\ (a, b) \in \{0, \ldots, p-1\}^2 \big\}.$$

Then $\mathcal{H}_2$ is a family of 2-wise independent hash functions. To check this we only need to evaluate, for any $x_1 \neq x_2$, i_1 and i_2,

$$\begin{aligned}\Pr_{a,b}(ax_1 + b = i_1 \wedge ax_2 + b = i_2) &= \Pr_{a,b}\left(a = \frac{i_2 - i_1}{x_2 - x_1} \wedge b = i_2 - x_2 a\right) \\ &= \frac{1}{p}\frac{1}{p},\end{aligned}$$

since a and b are chosen independently and uniformly at random.

When we study derandomization it will be important to construct many k-wise independent random variables using the fewest possible random bits. Here we have $|A| = p$ random variables, but the number of random bits is only what is required to choose a random $h \in \mathcal{H}$, so about $2 \log p$ bits. Compare this to $\log p$ bits needed to choose just *one* random value in $\{0, \ldots, p-1\}$!

1.4.2 Using pairwise independence

Now we can go back to the F_2 algorithm and modify it as follows: instead of using a completely random function h we use a random function $h : \{1, \ldots, n\} \to \{1, \ldots, n\}$ that is taken from a family $\mathcal{H}$ of 4-wise independent hash functions. You can easily check that the whole analysis goes through unchanged. But now the number of random bits required to choose h is only $O(\log n)$ (you will see an efficient construction of a family of 4-wise independent hash functions in your homework). To store the function, we simply store the random coins. Every time we need to evaluate $h(j)$ we read the random coins, recover the function, and evaluate it at the desired value. Thus for our algorithm to be time-efficient (and not only space-efficient) it is important that the evaluation of the function at a particular point can be done quickly, given the "raw" random coins. This is the case for the example of pairwise independent functions we saw earlier.

1.5 The median-of-means trick

Our main tool in the variance analysis of the streaming algorithms for estimating the frequency moments F_0 and F_2 has been Chebyshev's inequality, which is useful whenever we have good control over the variance $\mathbf{E}[(Z-\mu)^2]$ of Z. Using the inequality we were able to show that our algorithm gives an answer such that, first the answer is correct on expectation, and second it falls within a constant multiplicative factor of the right answer with constant probability. Now we'll see how to "boost" both the accuracy and success probability of any such algorithm, by running it multiple times.

First let's make a very simple observation. Suppose we execute any randomized procedure N times, using independent random coins at each time. Let the results be random variables $X_1, \ldots, X_N$, and set Z to be the average result, $Z = (1/N)(X_1 + \cdots + X_N)$. Then

$$\mathbf{E}[Z] = \frac{1}{N}\big(\mathbf{E}[X_1] + \cdots + \mathbf{E}[X_N]\big) = \frac{1}{N}\big(\mu_1 + \cdots + \mu_N\big),$$

and

$$\begin{aligned}
\mathbf{Var}(Z) &= \mathbf{E}\left[\left(\frac{1}{N}\big(X_1 + \cdots X_N\big) - \frac{1}{N}\big(\mu_1 + \cdots + \mu_N\big)\right)^2\right] \\
&= \frac{1}{N^2}\sum_i \mathbf{E}\left[\big(X_i - \mu_i\big)^2\right] - \frac{2}{N^2}\sum_{i,j} \mathbf{E}\left[(X_i - \mu_i)(X_j - \mu_j)\right] \\
&= \frac{1}{N^2}\sum_i \mathbf{Var}(X_i),
\end{aligned}$$

where in the last line we used that X_i and X_j are independent for $i \neq j$ to write $\mathbf{E}[X_i X_j] = \mathbf{E}[X_i]\,\mathbf{E}[X_j]$. This shows that taking the average of multiple independent runs decreases variance linearly, with a corresponding improvement in accuracy for our algorithm. Moreover, notice how once again the only property we need is really just pairwise independence.

Therefore, this method of amplification would work even if we didn't use completely independent functions h for each executation of the algorithm, letting us save once again on the random bits.

But suppose now that we can afford full independence, can we get a more efficient amplification of the success probability? We will see how this can be done by using a powerful concentration bound for sums of independent random variables, the Chernoff bound.

Theorem 1.6. *For any $\varepsilon, \delta > 0$ let*

$$t = C \log \frac{1}{\delta} \qquad \text{and} \qquad k = 3\frac{\mathbf{Var}(X)}{\varepsilon^2 \mathbb{E}[X]^2},$$

where C is some universal constant. Let X_{ij}, for $i \in \{1, \ldots, t\}$ and $j \in \{1, \ldots, k\}$, be independent random variables with the same distribution as X. Let

$$Z = \text{median}_{i \in \{1,\ldots,t\}} \Big(\frac{1}{k} \sum_{j=1}^{k} X_{ij} \Big).$$

Then $\mathbb{E}[Z] = \mu$ and $\mathbf{Pr}(|Z - \mu| \geq \varepsilon\mu) \leq \delta$.

Note that the number of copies required to drive the probability of error below δ scales as $\log(1/\delta)$, and not $1/\delta$ as it would if we were to rely only on Chebyshev's inequality.

Proof. Let $Y_i = \frac{1}{k} \sum_j X_{ij}$ for each $i \in \{1, \ldots, t\}$. Using linearity of expectation, $\mathbb{E}[Y_i] = \mu$, and using independence,

$$\mathbf{Var}(Y_i) = \frac{1}{k^2} \sum_j \mathbf{Var}(X_{ij}) = \frac{\mathbf{Var}(X)}{k}.$$

Applying Chebyshev's inequality, for each i,

$$\mathbf{Pr}(|Y_i - \mu| \geq \varepsilon\mu) \leq \frac{\mathbf{Var}(Y_i)}{\varepsilon^2\mu^2} = \frac{\mathbf{Var}(X)}{k\varepsilon^2\mathbb{E}[X]^2} = \frac{1}{3}.$$

For each i let W_i be a random variable that is 1 if $|Y_i - \mu| \geq \varepsilon\mu$. Then by the above bound $\mathbb{E}[W_i] \leq 1/3$, and $|Z - \mu| \geq \varepsilon\mu$ only if $W = \sum W_i > t/2$. Applying Chebyshev's inequality,

$$\begin{aligned}
\mathbf{Pr}\Big(\sum_{i=1}^{n} W_i > \frac{t}{2} \Big) &\leq \mathbf{Pr}\Big(\Big| \sum_{i=1}^{n} W_i - \mathbf{E}\Big[\sum_{j=1}^{n} W_j \Big] \Big| > \frac{t}{6} \Big) \\
&\leq \frac{t\,\mathbf{Var}(W_1)}{(t/6)^2} \\
&\leq \frac{1}{3}\frac{36}{t},
\end{aligned}$$

since $\mathbf{Var}(W_1) \leq \mathbf{E}[W_1^2] = \mathbf{E}[W_1] \leq 1/3$. To make this bound less than δ it is sufficient to take $t = 12/\delta$. But the theorem claims much better, $t = \log(1/\delta)$! That this t is enough is a consequence of the Chernoff bound, that we will see next. □

1.6 Chernoff Bounds

In the kind of scenario from the previous example, where $W = W_1 + \cdots + W_n$ is the sum of independent random variables, it is possible to do much better than Markov or Chebyshev. Chebyshev's inequality takes into account information about the variance, and it only requires the W_i to be pairwise independent. The Chernoff bound will do better by looking at all higher-order moments $\mathbb{E}[W^k]$ simultaneously, and using full independence.

Here is how we do it. For any $t \geq 0$ we can write

$$\begin{aligned}
\mathbf{Pr}\left(W \geq (1+\delta)\mu\right) &= \mathbf{Pr}\left(e^{tW} \geq e^{t(1+\delta)\mu}\right) && (x \mapsto e^x \text{is non-negative increasing}) \\
&\leq \frac{\mathbb{E}[e^{tW}]}{e^{t(1+\delta)\mu}} && \text{(Markov's inequality)} \\
&= e^{-t(1+\delta)\mu} \prod_{i=1}^{n} \mathbb{E}[e^{tW_i}] && \text{(independence)} \\
&= e^{-t(1+\delta)\mu} \prod_{i=1}^{n} \Big(p_i e^t + (1-p_i)1\Big) && \text{(for Boolean } W_i\text{)} \\
&\leq e^{-t(1+\delta)\mu} \prod_{i=1}^{n} e^{p_i(e^t-1))} && \text{(Taylor series: } 1+x \leq e^x\text{)} \\
&= e^{(\mu(e^t-1)-t(1+\delta)\mu)}.
\end{aligned}$$

Now we solve for t to find the best possible bound. If we take the derivative of the exponential term with respect to t and set it to 0, we find that the RHS has a minimum at $\mu e^t - (1+\delta)\mu = 0$, i.e. $t = \ln(1+\delta)$. This gives us the final bound

$$\mathbf{Pr}\left(W \geq (1+\delta)\mu\right) \leq \Big(\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\Big)^\mu.$$

A similar proof can be done for $W \leq (1-\delta)\mu$.

Exercise 3. In the fourth step above we used the fact that $W_i \in \{0,1\}$. Suppose now we only assume that $W_i \in [0,1]$, with $\mathbf{E}[W_i] = \mu/n$. How does that step need to be updated?

Assuming the result of the exercise, we have proved the following:

Theorem 1.7. *(Multiplicative Chernoff Bound] For any independent random variables $X_1, \ldots, X_n$ with $X_i \in (0,1]$ and $Z = \sum_{i=1}^{n} X_i$, $\mu = \mathbb{E}[Z]$:*

$$\mathbf{Pr}\left(Z \geq (1+\delta)\mu\right) \leq \Big(\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\Big)^\mu, \qquad \mathbf{Pr}\left(Z \leq (1-\delta)\mu\right) \leq \Big(\frac{e^{-\delta}}{(1-\delta)^{(1-\delta)}}\Big)^\mu.$$

Exercise 4. Show that for $\delta \in (0,1]$ the Chernoff bound implies the following weaker but often more convenient form:

$$\mathbf{Pr}\left(Z \geq (1+\delta)\mu\right) \leq e^{-\frac{\delta^2\mu}{3}}, \qquad \mathbf{Pr}\left(Z \leq (1-\delta)\mu\right) \leq e^{-\frac{\delta^2\mu}{2}}.$$

We can now finish the analysis of the median-of-means trick:

End of proof of Theorem 1.6. Applying the Chernoff bound to the W_i,

$$\mathbf{Pr}\left(\sum_{i=1}^{t} W_i > \frac{t}{2}\right) \leq e^{-\frac{(1/2)^2(t/3)}{3}} = e^{-\frac{t}{36}} \leq \delta$$

provided the constant C from the theorem is chosen large enough. □

1.7 Applications

We will multiple applications of the Chernoff bound. The first one is to a geometric problem: reducing the dimension of high-dimensional data. It will require us to prove a variant of the bound that is adapted to Gaussian (instead of Bernoulli) random variables, and will prove very useful in future algorithmic applications. The second application is to a purely combinatorial problem: counting the maximum load of a bin when balls are allocated (almost) randomly.

1.7.1 The Johnson-Lindenstrauss lemma

Suppose given a set of data points $x_1, \ldots, x_n \in \mathbb{R}^d$, where we think of d as being very large. For example, the x_i are "feature vectors": each of the d coordinates of x_i contains a numerical value for an attribute associated to the i-th object in our dataset. Suppose we are only interested in the rough geometry of this set — that is, we care about pairwise distances $\|x_i - x_j\|^2$, a good measure of similarity between elements of our dataset. We might also want to compute the distances $\|y - x_i\|^2$, where y is a new element. But d is very large. Is there a way to provide a "low-dimensional sketch" of our dataset that would capture its geometry, at least in an approximate sense. Note that we can already assume $d \leq n$. This is because the linear span of n vectors has dimension at most n, so we don't need more than n dimensions to represent the vectors. The Johnson-Lindenstrauss (JL) lemma shows that, if we are willing to allow for a small approximation error in the distances, we can go much lower — essentially, $\log(n)$ dimensions. More generally, the JL dimensionality reduction lemma provides a powerful technique for solving high-dimensional problems such as:

- Proximity problems: nearest neighbor, closest/furthest pair, Euclidean minimum spanning tree;
- Clustering, information retrieval;
- Learning an unknown mixture of Gaussians;
- Dimensionality reduction for online settings, such as sketching for streaming with limited storage;

and many more; in fact there is even a whole book devoted to the topic: "The Random Projection Method", by Santosh Vempala. Before stating and proving the JL lemma we introduce a little background on Gaussian random variables.

Gaussian random variables

A real continuous random variable X is defined by a nonnegative, integrable density function $\gamma : \mathbb{R} \to \mathbb{R}_+$ such that

$$\mathbf{Pr}\left(X \in [a,b]\right) = \int_a^b \gamma(x)\,dx.$$

Definition 1.8. We say X is *Gaussian*, or normally distributed, with mean μ and variance σ^2 when

$$\gamma(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2}, \tag{1.1}$$

in which case we write $X \sim \mathcal{N}(\mu, \sigma^2)$. We further say X is *standard Gaussian* if $\mu = 0$ and $\sigma^2 = 1$.

Here are some standard but useful properties of the Gaussian distribution.

Lemma 1.9.

(1) The density γ defined in (1.1) is normalized, i.e.

$$\int_{-\infty}^{\infty} \gamma(x)\,dx = 1.$$

(2) Let $Z = c_1X_1 + c_2X_2$ where $X_1, X_2 \sim \mathcal{N}(0,1)$ are independent random variables. Then $Z \sim \mathcal{N}(0, c_1^2 + c_2^2)$.

(3) Let $X \sim \mathcal{N}(0,1)$ and $t < 1/2$. Then

$$\mathbf{E}[e^{tX^2}] = \frac{1}{\sqrt{1-2t}}.$$

Exercise 5. Prove the lemma. For 1., show that it is enough to check normalization for $\mu = 0, \sigma^2 = 1$. If $I = \int_{-\infty}^{\infty} e^{-x^2/2}\,dx$, first verify that I is well-defined (it should be bounded) and then show that $I^2 = 2\pi$ by performing a change of variables. For 2., write the integral for the cumulative distribution function $\mathbf{Pr}(Z \le t)$ and again use rotation symmetry.

The Johnson-Lindenstrauss lemma

The JL lemma addresses the question of low-dimensional embeddings of points in Euclidean space that approximately preserve distances. The construction we give not only shows existence, but also yields a randomized algorithm for finding a linear embedding satisfying the desired properties with high probability. The lemma can be stated as follows.

Lemma 1.10 (Johnson-Lindenstrauss)**.** *For any integer $d > 0$, $0 < \varepsilon, \delta < 1/2$ and integer $k > 4\ln(2/\delta)/(\varepsilon^2 - \varepsilon^3)$ there exists a distribution on $k \times d$ real matrices G such that for any $x \in \mathbb{R}^d$,*

$$\mathbf{Pr}\left((1-\varepsilon)\|x\|^2 \le \|Gx\|^2 \le (1+\varepsilon)\|x\|^2\right) > 1 - \delta.$$

Proof. The distribution we use is simple: $G \in \mathbb{R}^{k\times d}$ is formed by choosing coefficients $G_{ij} \sim \mathcal{N}(0, 1/k)$ i.i.d. Let

$$Z = \frac{\|Gx\|^2}{\|x\|^2} = \sum_{i=1}^{k} \frac{(Gx)_i^2}{\|x\|^2},$$

so that the desired bounds are $(1-\varepsilon) \le Z \le (1+\varepsilon)$ w.h.p. Observe that, for all i, $(Gx)_i = \sum_{j=1}^{d} G_{ij}x_j \sim \mathcal{N}(0, \|x\|^2/k)$, by the second item from Lemma 1.9 and our choice of G_{ij}, so

$$\mathbf{E}[(Gx)_i^2] = \mathbf{Var}[(Gx)_i] = \frac{\|x\|^2}{k},$$

and $\mathbf{E}[Z] = \frac{1}{\|x\|^2}\sum_{i=1}^{k}\frac{\|x\|^2}{k} = 1$.

Now we want a high probability concentration bound on Z. Let $X_i = \frac{(Gx)_i}{\|x\|}$, so that $Z = \sum_{i=1}^{k} X_i^2$. Since the rows of G are independent it follows that the X_i are independent, so Z is a sum of i.i.d. random variables. Unfortunately these random variables are not bounded, so we can't directly apply the Chernoff bound, or generalizations such as Bernstein's inequality. Instead we go back to the basics and apply the Laplace transform method from scratch. This will also give us a sharper bound.

We bound $\mathbf{Pr}(Z \ge (1+\varepsilon))$, the other tail being analogous.

$$\begin{aligned}
\mathbf{Pr}(Z \ge (1+\varepsilon)) &= \mathbf{Pr}\left(e^{tkZ} \ge e^{tk(1+\varepsilon)}\right) \\
&\le \frac{\mathbf{E}[e^{tkZ}]}{e^{tk(1+\varepsilon)}} \\
&= \frac{\prod_{i=1}^{k}\mathbf{E}[e^{tkX_i^2}]}{e^{tk(1+\varepsilon)}} \\
&= \frac{1}{(1-2t)^{k/2}e^{tk(1+\varepsilon)}},
\end{aligned}$$

where the second equality uses independence of the X_i and the third equality follows from item 3. in Lemma 1.9 (using $\sqrt{k}X_i \sim \mathcal{N}(0,1)$ and assuming $t < 1/2$). Now we pick $t = \frac{\varepsilon}{2(1+\varepsilon)} < 1/2$, so that this upper bound simplifies to

$$\begin{aligned}
\left((1+\varepsilon)e^{-\varepsilon}\right)^{k/2} &\le ((1+\varepsilon)(1-\varepsilon+\varepsilon^2/2))^{k/2} \\
&= (1-\varepsilon^2/2+\varepsilon^3/2)^{k/2} \\
&\le e^{-(\varepsilon^2-\varepsilon^3)k/4},
\end{aligned}$$

where the inequalities follow by Taylor expansion. Choosing $k > 4\ln(2/\delta)/(\varepsilon^2-\varepsilon^3)$ yields a tail bound of at most $\delta/2$. Adding the bound for the lower tail yields an overall probability of failure of at most δ. □

The following is a simple but important consequence:

Theorem 1.11. *Let P be a set of n points in $\mathbb{R}^d$ and $0 < \varepsilon < 1$. For dimension $k > \frac{8 \ln n}{\varepsilon^2 - \varepsilon^3}$, there exists a linear map $\varphi : \mathbb{R}^d \to \mathbb{R}^k$, such that, for all $u, v \in P$,*

$$(1-\varepsilon)\|u-v\|^2 \leq \|\varphi(u) - \varphi(v)\|^2 \leq (1+\varepsilon)\|u-v\|^2,$$

that is, the mapping φ does not distort distances too much.

Proof. The construction for φ is simply to choose G randomly as in Lemma 1.10, and set $\varphi(u) = Gu$. Then for every pair (u, v), $\|\varphi(u) - \varphi(v)\|^2 = \|G(u-v)\|^2$. If we choose $\delta = O(1/n^2)$ in Lemma 1.10 we get that for any pair (u, v) the probability that $(1-\varepsilon)\|u - v\|^2 < \|G(u-v)\|^2 < (1+\varepsilon)\|G(u+v)\|^2$ is at least $1 - 1/(2n^2)$. Taking a union bound over all possible pairs, a random G will work with probability at least $1/2$, and in particular there exists a G, hence a φ, that works. □

Remark 1.12. We can tweak the constant factor in the threshold for k to get a high probability bound that φ works, e.g., choosing $k > \frac{12 \ln n}{\varepsilon^2 - \varepsilon^3}$ yields failure probability at most $\binom{n}{2}\frac{2}{n^3} < \frac{1}{n}$. Such a bound yields a Las Vegas algorithm for constructing a working φ (repeatedly sample a φ and check the distortion of φ until success).

Remark 1.13. A somewhat cleaner threshold for k of the form $k = \Omega(\varepsilon^{-2} \ln n)$ suffices if we have an upper bound on ε, e.g., if $\varepsilon < 1/2$, then $\varepsilon^2 - \varepsilon^3 > \varepsilon^2/2$ so

$$k > \frac{16 \ln n}{\varepsilon^2} \implies k > \frac{8 \ln n}{\varepsilon^2 - \varepsilon^3}.$$

1.7.2 Balanced allocation

Suppose there are n servers and n jobs. Our goal is to assign jobs to servers in a balanced way. What is an easy way to do this? Suppose we just assign jobs at random. Will the allocation be balanced — how many jobs will go to the server with the heaviest load?

Random Allocation

The problem is equivalent to assigning n balls to n bins in a way that the load of the bin with the most balls is minimized. So suppose that each ball is dropped in a random bin. Let Z_i count the number of balls in bin i: $Z_i = \sum_j X_{ij}$, where $X_{ij} = 1$ if ball j falls into bin i, and $X_{ij} = 0$ otherwise. Then

$$\mathbf{E}[Z_i] = \mathbf{E}\left[\sum_{j=1}^{n} X_{ij}\right] = \sum_{j=1}^{n} \mathbf{E}[X_{ij}] = \sum_{j=1}^{n} \frac{1}{n} = 1.$$

Using the Chernoff bound, for any $k \geq 1$

$$\mathbf{Pr}\left(Z_i \geq k\right) = \mathbf{Pr}\left(Z_i \geq k\,\mathbf{E}[Z_i]\right) \leq \left(\frac{e^{k-1}}{k^k}\right)^{\mathbf{E}[Z_i]} = \frac{e^{k-1}}{k^k}.$$

Let $k = \frac{3 \ln n}{\ln \ln n}$. The reason for this choice will become clear soon. Applying the union bound, the probability that any bin has at least k balls is at most

$$\begin{aligned}
\mathbf{Pr}\left(\text{any bin} \geq \frac{3 \ln n}{\ln \ln n}\right) &\leq n\Big(\frac{e \ln \ln n}{3 \ln n}\Big)^{\frac{3 \ln n}{\ln \ln n}} \\
&= n \exp\Big(\frac{3 \ln n}{\ln \ln n}\Big(1 + \ln\big(\frac{\ln \ln n}{3 \ln n}\big)\Big)\Big] \\
&\leq n \exp\Big(\frac{3 \ln n}{\ln \ln n}\Big(\ln \ln \ln n - \ln \ln n\Big)\Big] \\
&= n \exp\Big(\frac{3 \ln n \cdot \ln \ln \ln n}{\ln \ln n} - 3 \ln n\Big).
\end{aligned}$$

For large n, $\ln \ln \ln n \ll \ln \ln n$, and so

$$\mathbf{Pr}\left(\text{any bin} \geq \frac{3 \ln n}{\ln \ln n}\right) \leq n \exp(-2 \ln n) = \frac{1}{n} \tag{1.2}$$

The motivation for the choice of k is now clear. For $k = \frac{3 \ln n}{\ln \ln n}$ and large n, the probability that there is a bin with many balls is small and decreases with n. It is possible to show that this is tight: with high probability there will always exist a bin that contains $\Omega(\ln n / \ln \ln n)$ balls.

The Power of Two Choices

How can we achieve a more balanced allocation? Here is a simple trick we could try to play: for each ball, pick *two* bins at random, and drop the ball in the bin with fewer balls (to make the analysis simpler we'll allow that the two bins happen to be the same, in which case we have no choice).

Let B_i be the random variable that counts the number of bins with at least i balls, after all n balls have been distributed. Let β_i be an upper bound on B_i: $B_i \leq \beta_i$. Suppose that at the beginning of the t-th step there are T_i bins having at least i balls each. What is the probability that the t-th ball is placed in a bin having at least i balls? For this to happen both bins selected need to have at least i balls. Then

$$\mathbf{Pr}\big(\text{ball } t \text{ placed in bin with} \geq \text{i balls}\big) = \Big(\frac{T_i}{n}\Big)^2 \leq \frac{B_i^2}{n^2} \leq \frac{\beta_i^2}{n^2}.$$

So the expected number of bins containing at least $i+1$ balls is at most the expected number of successes in a Bernoulli experiment with n trials and probability of success $\frac{\beta_i^2}{n^2}$:

$$\mathbf{E}[B_{i+1}] \leq n \cdot \frac{\beta_i^2}{n^2} = \frac{\beta_i^2}{n}.$$

Applying Markov's inequality, $\mathbf{Pr}(B_{i+1} \geq e\frac{\beta_i^2}{n}) \leq e^{-1}$. This motivates the following sequence of β_i's. Trivially, $B_6 \leq \frac{n}{6} \leq \frac{n}{2e}$. So we set

$$\beta_6 = \frac{n}{2e},\ \beta_7 = \frac{n}{2^2 e},\ \beta_8 = \frac{n}{2^4 e}, \ldots, \beta_i = \frac{n}{2^{2^{i-6}} e}.$$

Since $\mathbf{Pr}\left(B_{i+1} \geq e\frac{\beta_i^2}{n}\right)$ is low, the probability that $B_i \leq \beta_i$ for each i is high. Let E_i be the event that $B_i \leq \beta_i$. Note that $\mathbf{Pr}(E_6) = 1$.

Lemma 1.14. *For all i s.t. $\beta_i^2 \geq 2n \ln n$, $\mathbf{Pr}(\neg E_{i+1}|E_i) \leq \frac{1}{n^2 \mathbf{Pr}(E_i)}$.*

Proof.

$$\begin{aligned}
\mathbf{Pr}(\neg E_{i+1}|E_i) &= \frac{\mathbf{Pr}(\neg E_{i+1} \wedge E_i)}{\mathbf{Pr}(E_i)} \\
&\leq \frac{\mathbf{Pr}\left(\text{Binomial}\left(n, \frac{B_i^2}{n^2}\right) \geq \frac{e\beta_i^2}{n}\right)}{\mathbf{Pr}(E_i)} \\
&= \frac{\mathbf{Pr}\left(\text{Binomial}(n, \frac{B_i^2}{n^2}) \geq e\,\mathbf{E}(\text{Binomial}(n, \frac{B_i^2}{n^2}))\right)}{\mathbf{Pr}(E_i)}.
\end{aligned}$$

Here, we used that $\beta_{i+1} = \frac{e\beta_i^2}{n}$. $\neg E_{i+1}$ then occurs if $\text{Binomial}(n, \frac{B_i^2}{n^2}) \geq B_{i+1} > \frac{e\beta_i^2}{n}$. Now, use the Chernoff bound as $\mathbf{Pr}(Z \geq e\,\mathbf{E}(Z)) \leq \exp(-\mathbf{E}(Z))$ to deduce $\mathbf{Pr}(\neg E_{i+1}|E_i) \leq \frac{\exp\left(-\frac{\beta_i^2}{n}\right)}{\mathbf{Pr}(E_i)} \leq \frac{\exp(-2\ln n)}{\mathbf{Pr}(E_i)} \leq \frac{1}{n^2 \mathbf{Pr}(E_i)}$. □

Lemma 1.15. *For all i s.t. $\beta_i^2 \geq 2n \ln n$, $\mathbf{Pr}(\neg E_{i+1}) \leq \frac{i+1}{n^2}$.*

Proof. Use induction on i. The base case is $\mathbf{Pr}(\neg E_6) = 0 \leq \frac{7}{n^2}$. Next we have

$$\begin{aligned}
\mathbf{Pr}(\neg E_{i+1}) &= \mathbf{Pr}(E_i)\,\mathbf{Pr}(\neg E_{i+1}|E_i) + \mathbf{Pr}(\neg E_i)\,\mathbf{Pr}(\neg E_{i+1}|\neg E_i) \\
&\leq \mathbf{Pr}(E_i)\frac{1}{n^2\,\mathbf{Pr}(E_i)} + \frac{i}{n^2}\,\mathbf{Pr}(\neg E_{i+1}|\neg E_i) \\
&\leq \frac{1}{n^2} + \frac{i}{n^2} \\
&\leq \frac{i+1}{n^2},
\end{aligned}$$

where $\mathbf{Pr}(\neg E_i) \leq \frac{i}{n^2}$ comes from the induction hypothesis.

□

We have shown that for all $\beta_i^2 \geq 2n \ln n$, the probability that β_i does not bound B_i is low, and decreases like $O\left(\frac{1}{n^2}\right)$. Now, we must consider the other case where $\beta_i^2 < 2n \ln n$. Let i^* be the minimum i for which $\beta_i^2 < 2n \ln n$. Then, $i^* \leq \frac{\ln \ln n}{\ln 2} + O(1)$.

Lemma 1.16. $\mathbf{Pr}(B_{i^*+2} \geq 1) \leq O\left(\frac{\ln(n)^2}{n}\right)$.

Proof. Define $E_{i^*+1} = \{B_{i^*+1} \leq 6 \ln n\}$. Then

$$\begin{aligned}
\mathbf{Pr}(\neg E_{i^*+1}) &\leq \mathbf{Pr}(B_{i^*+1} \geq 6 \ln n | E_{i^*}) \, \mathbf{Pr}(E_{i^*}) + \mathbf{Pr}(\neg E_{i^*}) \\
&\leq \frac{\mathbf{Pr}(\text{Binomial}(n, \frac{2 \ln n}{n}) \geq 6 \ln n)}{\mathbf{Pr}(E_{i^*})} \cdot \mathbf{Pr}(E_{i^*}) + \frac{1}{n} \\
&\leq \frac{1}{n^2} + \frac{1}{n} \\
&= O\Big(\frac{1}{n}\Big).
\end{aligned}$$

Thus

$$\begin{aligned}
\mathbf{Pr}(B_{i^*+2} \geq 1) &\leq \mathbf{Pr}(B_{i^*+2} \geq 1 | E_{i^*+1}) \cdot \mathbf{Pr}(E_{i^*+1}) + \mathbf{Pr}(\neg E_{i^*+1}) \\
&\leq \frac{\mathbf{Pr}(\text{Binomial}(n, (6 \ln n/n)^2) \geq 1)}{\mathbf{Pr}(E_{i^*+1})} \cdot \mathbf{Pr}(E_{i^*+1}) + O\Big(\frac{1}{n}\Big) \\
&\leq \Big(\frac{6 \ln n}{n}\Big)^2 \cdot n + O\Big(\frac{1}{n}\Big) \\
&= O\Big(\frac{(\ln n)^2}{n}\Big),
\end{aligned}$$

as desired.

□

1.7.3 Fast Johnson-Lindenstrauss

The Johnson-Lindenstrauss dimension reduction lemma has many algorithmic applications, and we will see some of them today. Typically, the lemma is used to speed up an algorithm by first, applying the dimension reduction map to reduce the dimension of the problem to some small k, and second, solving the low-dimensional problem. For this to be effective it is important that the map can be applied efficiently. Using the construction from the previous lecture, where the dimension reduction matrix was a $k \times d$ matrix G with entries i.i.d. Gaussian, computing the image of a vector $x \in \mathbb{R}^d$ will take time about kd. In some applications x might be a sparse vector, in which case the running time would be $k\|x\|_0$, where $\|x\|_0$ is the number of non-zero entries of x. If $\|x\|_0$ is a constant, this still depends on k, which depending on the problem could be large.

In order to do better, we want to show that the JL lemma holds for some random matrices G that are more structured, and such that Gx can be computed very efficiently. Today we will see that in fact we can choose G to have many entries equal to to 0: each column of G will have only s nonzero entries, where $s = \Theta(\varepsilon^{-1} \log(1/\delta))$ is independent of d and k. We'll see that we can get exactly (up to constant factors) the same guarantees as with the completely random G, but now the computation time for Gx is much faster: it no longer depends on k!

Consider the following construction for a $k \times d$ matrix G. Each column of G is split into s contiguous blocks of size k/s. In each block there will be a single non-zero entry. We use

an indicator variable $\eta_{r,i} = 1$ if the r-th entry of the i-th column is nonzero, and $\eta_{r,i} = 0$ otherwise. The nonzero entries will be $\sigma_{r,i}/\sqrt{s}$, where $\sigma_{r,i} \in \{\pm 1\}$ are chosen uniformly at random. Thus we define

$$G_{r,i} = \frac{\eta_{r,i}\sigma_{r,i}}{\sqrt{s}}$$

for every $r \in \{1, \dots, k\}$ and $i \in \{1, \dots, d\}$. Before we go on let's verify that this is at least good in expectation:

$$\begin{aligned}
\mathbf{E}\left[\|Gx\|^2\right] &= \mathbf{E}\left[\frac{1}{s}\sum_{r=1}^{k}\sum_{i,j=1}^{d}\eta_{r,i}\eta_{r,j}\sigma_{r,i}\sigma_{r,j}x_i x_j\right] \\
&= \frac{1}{s}\sum_{r=1}^{k}\sum_{i=1}^{d}\eta_{r,i}^2 x_i^2 \\
&= \sum_{i=1}^{d} x_i^2 = \|x\|^2,
\end{aligned}$$

where for the second equality we used $\mathbf{E}[\sigma_{r,i}] = 0$ and $\sigma_{r,i}^2 = 1$, and for the third we used that there are exactly s nonzero $\eta_{r,i}$ per column so $\sum_{r=1}^{k}\eta_{r,i} = s$.

We have not yet specified how the locations $\eta_{r,i}$ of the nonzero entries are chosen. There is only one important condition that these entries must satisfy: we need that there are few collisions between non-zero entries in different columns. More precisely we will require that for any $i \neq j \in \{1, \dots, d\}$,

$$\sum_{r=1}^{k}\eta_{r,i}\eta_{r,j} = O\Big(\frac{s^2}{k}\Big). \tag{1.3}$$

Note that if we chose the nonzero locations uniformly at random within each block then the expected number of collisions per pair of columns is precisely $s \times (s/k) = s^2/k$. Using a Chernoff bound, the probability that there are more than e.g. $2s^2/k$ collisions is exponentially small in s^2/k, so if $s^2/k = \Omega(\log(d/\delta))$ we can apply a union bound and the probability that any pair of columns has more than $2s^2/k$ collisions will be at most δ. So a random construction works, provided the sparsity s, and hence the final computation time, depends logarithmically on d. In fact it is possible to remove this requirement by using a finer analysis (based on a weaker assumption than (1.3)), but we will not show this in this lecture. We're going to cheat a little bit, assume we have a way to choose the $\{\eta_{r,i}\}$ such that (1.3) holds, and show the following:

Theorem 1.17 (Kane-Nelson)**.** *For any integer $d > 0$ and $0 < \varepsilon, \delta < 1/2$ the distribution on $k \times d$ real matrices G described above is such that if $k = \Omega(\varepsilon^{-2}\log(1/\delta))$ and $s = \Omega(\varepsilon^{-1}\log(1/\delta))$ then for any $x \in \mathbb{R}^d$,*

$$\mathbf{Pr}\left((1-\varepsilon)\|x\|^2 \leq \|Gx\|^2 \leq (1+\varepsilon)\|x\|^2\right) > 1 - \delta.$$

The main ingredient in the proof is a concentration bound due to Hanson and Wright that applies to quadratic forms: for B a real $n \times n$ matrix we are interested in studying $\sum_{i,j=1}^n B_{ij} z_i z_j$ where the $z_i \in \{\pm 1\}$ are random signs. The expectation of this is

$$\mathbf{E}\Big[\sum_{i,j=1}^n B_{ij} z_i z_j\Big] = \sum_i B_{ii} = \mathrm{Tr}(B).$$

The following theorem shows that the concentration properties of this expression are governed by two different norms of B: the operator norm $\|B\|$ (the largest singular value), and the Frobenius norm

$$\|B\|_F = \sqrt{\mathrm{Tr}(B^T B)} = \sqrt{\sum_{i,j=1}^n B_{i,j}^2}.$$

Theorem 1.18 (Hanson-Wright). *Let $z = (z_1, \dots, z_n)^T$ be a vector of i.i.d. Rademacher $\{\pm 1\}$ random variables. For any $B \in \mathbb{R}^{n \times n}$ and $p \geq 2$,*

$$\mathbf{E}\left|z^T B z - \mathrm{Tr}(B)\right|^p \leq C^p \max\left\{\sqrt{p}\|B\|_F, p\|B\|\right\}^p,$$

for some universal constant $C > 0$ independent of B, n, p.

Exercise 6. Show that the moment bound stated in Theorem 1.18 is equivalent to the following tail bound: there exists constants $C', C'' > 0$ such that for all $t > 0$,

$$\mathbf{Pr}\left(\left|z^T B z - \mathrm{Tr}(B)\right| > t\right) \leq C' e^{-C'' \min\left(\frac{t^2}{\|B\|_F^2}, \frac{t}{\|B\|}\right)}.$$

Proof of Theorem 1.17. We can write $(Gx)_r = \sum_j \eta_{r,j}\sigma_{r,j}x_j/\sqrt{s}$, and define

$$\begin{aligned} Z &= \|Gx\|^2 - 1 \\ &= \frac{1}{s}\sum_{r=1}^k \sum_{i,j} \eta_{r,i}\eta_{r,j}\sigma_{r,i}\sigma_{r,j}x_i x_j - 1 \\ &= \frac{1}{s}\sum_{r=1}^k \sum_{i \neq j} \eta_{r,i}\eta_{r,j}\sigma_{r,i}\sigma_{r,j}x_i x_j. \end{aligned}$$

Now the key observation is that we can rewrite this as $\sigma^T B \sigma$, where B is a block-diagonal matrix with k blocks, and each block is $d \times d$ with entries $\eta_{r,i}\eta_{r,j}x_i x_j$ for $i \neq j$, and 0 on the diagonal. In particular we have $\mathrm{Tr}(B) = 0$. To apply Hanson-Wright, we need to estimate the Frobenius norm and the operator norm of B. Let's start with the Frobenius norm:

$$\begin{aligned} \|B\|_F^2 &= \frac{1}{s^2}\sum_{i \neq j} x_i^2 x_j^2 \Big(\sum_{r=1}^k \eta_{r,i}\eta_{r,j}\Big) \\ &\leq O\Big(\frac{1}{k}\Big)\|x\|_2^4 \\ &= O\Big(\frac{1}{k}\Big), \end{aligned}$$

where the second line is by assumption (1.3).

Next we bound the operator norm. Since B is block-diagonal it suffices to bound the norm of any block r. We can write the r-th block $B_r = (S_r - D_r)/s$ where $S_r = (\eta_{r,i}\eta_{r,j}x_i x_j)_{i,j}$ and D_r is diagonal with coefficients $\eta_{r,i}^2 x_i^2$ on the diagonal. Now $\|D_r\| \leq \|x\|_\infty^2 \leq 1$ and $\|S_r\| = \|\eta_r x\|^2 \leq \|x\|^2 \leq 1$. Overall, $\|B\| \leq 2/s$.

Applying the (tail version of) the Hanson-Wright inequality,

$$\begin{aligned}\mathbf{Pr}\left(|Z| > \varepsilon\|x\|^2\right) &= \mathbf{Pr}\left(\left|\sigma^T B\sigma - \mathrm{Tr}(B)\right| > \varepsilon\right)\\ &\leq C' e^{-C\min\left(O(\varepsilon^2 k), O(\varepsilon s)\right)}\\ &\leq \delta,\end{aligned}$$

given the choice of s and k made in the theorem. □

1.7.4 Approximate nearest neighbors

Suppose given n points $x_1, \ldots, x_n \in \mathbb{R}^d$. Given a query $y \in \mathbb{R}^d$, which is the closest x_i to y? A typical algorithm for this problem will have two phases:

(1) *(Preprocessing)* Construct a data structure based on the n points.

(2) *(Query)* Given a new point $y \in \mathbb{R}^d$, query the data structure and return the index of a nearest neighbor.

If d is small, $d = 2$ or $d = 3$, there are some very efficient data structures for this problem, using quasi-linear (in n) space and answering each query in $O(\log n)$ time. But as soon as d grows (think $\log n \ll d \ll n$, for instance $d = n^{0.1}$) the problem suffers from the "curse of dimensionality": either the data structure needs to have size exponential in d, or the time per query must be linear in d.

For instance, the simplest algorithm would simply store the points x_i; given query y it evaluates all distances and returns the closest point. This requires $O(nd)$ space for the data structure and $O(nd)$ time per query. At the opposite end of the spectrum, we can construct a structure based on the Voronoi diagram of the x_i. This will require space $n^{O(d)}$, but only time $O(d \log n)$ per query; you can think of it as a spatial extension of the usual $O(\log n)$ binary search.

Let's see how we can do much better by using dimension reduction based on the Johnson-Lindenstrauss lemma. First, let's settle for ε-approximate nearest neighbors: given y, find i such that $\|y - x_i\| \leq (1+\varepsilon)\min_{1\leq j\leq n}\|y - x_j\|$. Suppose also that the closest point to y has distance 1 (we can always perform binary search to reduce to this case).

Fix a grid on $\mathbb{R}^d$ where each cube has sides of length $\varepsilon/\sqrt{d}$. For each i, let G_i be the set of grid cells that contain at least one point at distance at most 1 from x_i, and store all resulting grid cells in a hash table (where the key is an identification number for the cell, and the value is the index of the point x_i associated to that cell; if a cell has more than one x_i associated to it we can keep only one as the distances will necessarily be almost the

same). By a volume argument, the number of grid cells associated to any i is at most $(c/\varepsilon)^d$ for some constant c, so our data structure uses dn (to store the x_i) plus $O(n(c/\varepsilon)^d)$ space. Given a query y, we simply find the grid cell it is contained in and return the associated point i.

Ok, so we have $dn + O(n(c/\varepsilon)^d)$ space and $O(d)$ query time (to determine the cell and hash it; we can use a simple linear hash function $h(z_1, \ldots, z_d) = (a_1 z_1 + \cdots + a_d z_d \mod p) \mod s$), where p is prime and s is the hash table size). So this is very efficient in terms of query time, but still requires space exponential in d.

But now we can reduce the dimension! Apply the Johnson-Lindenstrauss lemma to project all the x_i to $d' = O(\varepsilon^{-2} \log n)$ dimensions. This gives $n^{O(\log(1/\varepsilon)/\varepsilon^2)}$ space. When a query y is made, we project it in $O(d\varepsilon^{-2} \log n)$ time, and answer the query in $O(d') = O(\varepsilon^{-2} \log n)$ time. For ε constant, we have space polynomial in N and query time $O(d \log n)$: this matches the query time of the basic Voronoi algorithm, but the space requirement is greatly reduced. Unfortunately the dependence on ε is rather bad. Nevertheless, this can be made into a practical algorithm as all operations are very simple, and it is used in practice. Note also the query time is dominated by the time required to compute the Johnson-Lindenstrauss embedding, and it is useful to reduce this: using the fast JL described earlier, if the queries x are sparse vectors we can get a query time that is independent of d and logarithmic in n.

1.7.5 Approximating matrix products

Suppose $A \in \mathbb{R}^{d \times n}$ and $B \in \mathbb{R}^{d \times m}$ are given. Naïvely the product $A^T B$ takes time $O(ndm)$ to compute. If $n = d = m$ we can do time $O(n^\omega)$, where $\omega \approx 2.373$, using fast matrix multiplication (Strassen's algorithm gives $\log_2 7 \approx 2.807$, and *much more work* gives small improvements); this can also be used to speed up the rectangular case by breaking up A, B in $d \times d$ blocks. Here we're going to see how to compute $A^T B$ approximately, with additive error $\varepsilon \|A\|_F \|B\|_F$ where $\|A\|_F = (\sum_{i,j} A_{i,j}^2)^{1/2}$ is the Frobenius norm. The idea is very simple: we insert a random low-dimensional projection $S \in \mathbb{R}^{d' \times d}$ and return the product $A^T S^T S B$. The whole product can be computed in time $O(ndd' + mdd' + nd'm)$, which is much better as long as $d' \ll n, d, m$.

One natural way to do this is via sampling: write $A^T B = \sum_{i=1}^d a_i b_i^T$ where a_i are the columns of a, and b_i the rows of B. Then we can try to approximate this sum by a random sample. Using a standard concentration bound you can check that to get additive error $\varepsilon \|A\|_F \|B\|_F$ with probability at least $1 - \delta$ it is enough to sample $\Omega(\varepsilon^{-2} \delta^{-1})$ terms. Moreover the dependence on δ can be improved to $\log(1/\delta)$ by using a standard amplification trick (repeat the experiment many times and output the median).

We'll see a way to achieve a similar guarantee using the Johnson-Lindenstrauss dimension reduction technique. Using the fast JL we saw earlier this lets us improve the dependence on ε from ε^{-2} to ε^{-1}. In fact we'll prove something slightly more general:

Theorem 1.19. *Let $\varepsilon, \delta \in (0, 1/2)$ and $\mathcal{D}$ a distribution on $d' \times d$ matrices such that for any*

unit vector $x \in \mathbb{R}^d$

$$\mathop{\mathbf{E}}_{S\sim\mathcal{D}} \left| \|Sx\|_2^2 - 1 \right|^\ell \leq \varepsilon^\ell \delta \tag{1.4}$$

for some $\ell \geq 2$. *Then for any* A, B *each having* d *rows,*

$$\mathop{\mathbf{Pr}}_{S\sim\mathcal{D}} \left(\|A^T S^T S B - A^T B\|_F > 3\varepsilon \|A\|_F \|B\|_F \right) < \delta.$$

Recall that in the last lecture we saw that if $\mathcal{D} = \mathcal{D}_{JL}$ is the Johnson-Lindenstrauss distribution then for any unit norm vector x

$$\mathop{\mathbf{Pr}}_{S\sim\mathcal{D}} \left(\left| \|Sx\|^2 - 1 \right| > \varepsilon \right) \leq e^{-\varepsilon^2 d'/8}.$$

Using the last problem from Homework 1 it follows that for any $\ell \geq 1$,

$$\mathbf{E} \left| \|S\|_x^2 - 1 \right|^\ell \leq (C\sqrt{\ell/d'})^\ell$$

for some constant C. So if we set $d' = \Omega(\log(1/\delta)\varepsilon^{-2})$ and $\ell = \log(1/\delta)$ we get that $\mathcal{D}_{JL}$ satisfies (1.4). The same is true for the sparse Johnson-Lindenstrauss transform we saw earlier.

Proof of Theorem 1.19. The proof is a good application of the moment method for proving concentration bounds. Fix $x, y \in \mathbb{R}^d$ with norm 1. We can write the inner product

$$(Sx)^T(Sy) = \frac{1}{2}\left(\|Sx\|_2^2 + \|Sy\|_2^2 - \|Sx - Sy\|_2^2 \right).$$

Recall the definition $\|X\|_\ell = (\mathbf{E}\,|X|^\ell)^{1/\ell}$ for a random variable X. Using the triangle inequality for $\|\cdot\|_\ell$ (this is also called *Minkowski's inequality*),

$$\begin{aligned}
\left\| (Sx)^T(Sy) - x^T y \right\|_\ell &= \frac{1}{2} \left\| (\|Sx\|_2^2 - 1) + (\|Sy\|_2^2 - 1) - (\|Sx - Sy\|_2^2 - \|x - y\|_2^2) \right\|_\ell \\
&\leq \frac{1}{2}\left(\left\| \|Sx\|_2^2 - 1 \right\|_\ell + \left\| \|Sy\|_2^2 - 1 \right\|_\ell - \left\| \|Sx - Sy\|_2^2 - \|x - y\|_2^2) \right\|_\ell \right) \\
&\leq \frac{1}{2}\left(\varepsilon\delta^{1/\ell} + \varepsilon\delta^{1/\ell} + \|x - y\|_2^2\, \varepsilon\delta^{1/\ell} \right) \\
&\leq 3\varepsilon\delta^{1/\ell}.
\end{aligned}$$

Let $x_1, \ldots, x_n$ be the columns of A and $y_1, \ldots, y_m$ the columns of B. Define a random variable

$$X_{i,j} = \frac{1}{\|x_i\|_2 \|y_j\|_2}\left((Sx_i)^T(Sy_j) - x_i^T y_j \right).$$

Then $\|A^T S^T SB - A^T B\|_F^2 = \sum_{i,j} \|x_i\|_2^2 \|y_j\|_2^2 X_{i,j}^2$. Using again the triangle inequality,

$$\begin{aligned}
\big\|\|A^T S^T SB - A^T B\|_F^2\big\|_{\ell/2} &= \Big\| \sum_{i,j} \|x_i\|_2^2 \|y_j\|_2^2 \, X_{i,j}^2 \Big\|_{\ell/2} \\
&\le \sum_{i,j} \|x_i\|_2^2 \|y_j\|_2^2 \, \|X_{i,j}^2\|_{\ell/2} \\
&= \sum_{i,j} \|x_i\|_2^2 \|y_j\|_2^2 \, \|X_{i,j}\|_\ell^2 \\
&\le (3\varepsilon\delta^{1/\ell})^2 \Big(\sum_{i,j} \|x_i\|_2^2 \|y_j\|_2^2 \Big) \\
&= (3\varepsilon\delta^{1/\ell})^2 \|A\|_F^2 \|B\|_F^2.
\end{aligned}$$

Finally by Markov's inequality,

$$\begin{aligned}
\mathbf{Pr}\left(\|A^T S^T SB - A^T B\|_F > 3\varepsilon \|A\|_F \|B\|_F \right) &\le \Big(\frac{1}{3\varepsilon \|A\|_F \|B\|_F} \Big)^\ell \mathbf{E} \, \|A^T S^T SB - A^T B\|_F^\ell \\
&\le \delta.
\end{aligned}$$

□