Class notes for Randomized Algorithms

Sariel Har-Peled[①]

December 10, 2019

[①]Department of Computer Science; University of Illinois; 201 N. Goodwin Avenue; Urbana, IL, 61801, USA; sariel@illinois.edu; http://sarielhp.org/. Work on this paper was partially supported by a NSF CAREER award CCR-0132901.

Contents

Chapter 1

Expectation and Quick Sort

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
December 10, 2019

Everybody knows that the dice are loaded
Everybody rolls with their fingers crossed
Everybody knows the war is over
Everybody knows the good guys lost
Everybody knows the fight was fixed
The poor stay poor, the rich get rich
That's how it goes
Everybody knows

Everybody knows, Leonard Cohen

1.1. Basic probability

Here we recall some definitions about probability. The reader already familiar with these definition can happily skip this section.

1.1.1. Formal basic definitions: Sample space, σ-algebra, and probability

A ***sample space*** Ω is a set of all possible outcomes of an experiment. We also have a set of events $\mathcal{F}$, where every member of $\mathcal{F}$ is a subset of Ω. Formally, we require that $\mathcal{F}$ is a σ-algebra.

Definition 1.1.1. A single element of Ω is an ***elementary event*** or an ***atomic event***.

Definition 1.1.2. A set $\mathcal{F}$ of subsets of Ω is a ***σ-algebra*** if:
(i) $\mathcal{F}$ is not empty,
(ii) if $X \in \mathcal{F}$ then $\overline{X} = (\Omega \setminus X) \in \mathcal{F}$, and
(iii) if $X, Y \in \mathcal{F}$ then $X \cup Y \in \mathcal{F}$.
More generally, we require that if $X_i \in \mathcal{F}$, for $i \in \mathbb{Z}$, then $\cup_i X_i \in \mathcal{F}$. A member of $\mathcal{F}$ is an ***event***.

As a concrete example, if we are rolling a dice, then $\Omega = \{1, 2, 3, 4, 5, 6\}$ and $\mathcal{F}$ would be the power set of all possible subsets of Ω.

Definition 1.1.3. A ***probability measure*** is a mapping $\mathbb{P} : \mathcal{F} \to [0, 1]$ assigning ***probabilities*** to events. The function $\mathbb{P}$ needs to have the following properties:
(i) Additive: for $X, Y \in \mathcal{F}$ disjoint sets, we have that $\mathbb{P}\left[X \cup Y\right] = \mathbb{P}\left[X\right] + \mathbb{P}\left[Y\right]$, and
(ii) $\mathbb{P}[\Omega] = 1$.

Definition 1.1.4. A ***probability space*** is a triple $(\Omega, \mathcal{F}, \mathbb{P})$, where Ω is a sample space, $\mathcal{F}$ is a σ-algebra defined over Ω, and $\mathbb{P}$ is a probability measure.

Definition 1.1.5. A ***random variable*** f is a mapping from Ω into some set $\mathcal{G}$. We require that the probability of the random variable to take on any value in a given subset of values is well defined. Formally, for any subset $U \subseteq \mathcal{G}$, we have that $f^{-1}(U) \in \mathcal{F}$. That is, $\mathbb{P}[f \in U] = \mathbb{P}\left[f^{-1}(U)\right]$ is defined.

Going back to the dice example, the number on the top of the dice when we roll it is a random variable. Similarly, let X be one if the number rolled is larger than 3, and zero otherwise. Clearly X is a random variable.

We denote the ***probability*** of a random variable X to get the value x, by $\mathbb{P}[X = x]$ (or sometime $\mathbb{P}[x]$, if we are lazy).

1.2. Expectation and conditional probability

1.2.1. Expectation

Definition 1.2.1 (Expectation). The expectation of a random variable X, is its average. Formally, the ***expectation*** of X is

$$\mathbb{E}\left[X\right] = \sum_x x\, \mathbb{P}\left[X = x\right].$$

Lemma 1.2.2 (Linearity of expectation). **Linearity of expectation** *is the property that for any two random variables X and Y, we have that* $\mathbb{E}\left[X + Y\right] = \mathbb{E}\left[X\right] + \mathbb{E}\left[Y\right]$.

Proof: $\mathbb{E}\left[X + Y\right] = \sum_{\omega \in \Omega} \mathbb{P}[\omega]\big(X(\omega) + Y(\omega)\big) = \sum_{\omega \in \Omega} \mathbb{P}[\omega] X(\omega) + \sum_{\omega \in \Omega} \mathbb{P}[\omega] Y(\omega) = \mathbb{E}\left[X\right] + \mathbb{E}\left[Y\right]$. ■

Example 1.2.3. Let F be a boolean formula with n variables in CNF form, with m clauses, where each clause has exactly k literals. A random assignment for F, where value 0 or 1 is picked with probability $1/2$, satisfies in expectation $(1 - 2^{-k})m$ of the clauses.

1.2.2. Conditional probability

Definition 1.2.4 (Conditional Probability). The ***conditional probability*** of X given Y, is the probability that $X = x$ given that $Y = y$. We denote this quantity by $\mathbb{P}[X = x \mid Y = y]$.

One useful way to think about the conditional probability $\mathbb{P}[X \mid Y]$ is as a function, between the given value of Y (i.e., y), and the probability of X (to be equal to x) in this case. Since in many cases x and y are omitted in the notation, it is somewhat confusing.

The conditional probability can be computed using the formula

$$\mathbb{P}[X = x \mid Y = y] = \frac{\mathbb{P}\left[(X = x) \cap (Y = y)\right]}{\mathbb{P}[Y = y]}.$$

For example, let us roll a dice and let X be the number we got. Let Y be the random variable that is true if the number we get is even. Then, we have that

$$\mathbb{P}\left[X = 2 \,\middle|\, Y = true\right] = \frac{1}{3}.$$

Definition 1.2.5. Two random variables X and Y are ***independent*** if $\mathbb{P}[X = x \mid Y = y] = \mathbb{P}[X = x]$, for all x and y.

Observation 1.2.6. *If X and Y are independent then $\mathbb{P}\big[X = x \,\big|\, Y = y\big] = \mathbb{P}[X = x]$ which is equivalent to $\dfrac{\mathbb{P}[X = x \cap Y = y]}{\mathbb{P}[Y = y]} = \mathbb{P}[X = x]$. That is, X and Y are* independent*, if for all x and y, we have that*

$$\mathbb{P}\big[X = x \cap Y = y\big] = \mathbb{P}\big[X = x\big]\,\mathbb{P}\big[Y = y\big].$$

Remark. Informally, and not quite correctly, one possible way to think about conditional probability $\mathbb{P}[X = x \mid Y = y]$ is as measuring the benefit of having more information. If we know that $Y = y$, do we have any change in the probability of $X = x$?

Lemma 1.2.8. *If X and Y are two random independent variables, then $\mathbb{E}[XY] = \mathbb{E}[X]\,\mathbb{E}[Y]$.*

1.2.3. Application: Approximating k-SAT

We remind the reader that an instance of 3SAT is a boolean formula, for example $F = (x_1 + x_2 + x_3)(x_4 + \overline{x_1} + x_2)$, and the decision problem is to decide if the formula has a satisfiable assignment. Interestingly, we can turn this into an optimization problem.

Max 3SAT

Instance: A collection of clauses: $C_1, \ldots, C_m$.
Question: Find the assignment to $x_1, \ldots, x_n$ that satisfies the maximum number of clauses.

Clearly, since 3SAT is NP-Complete it implies that Max 3SAT is NP-Hard. In particular, the formula F becomes the following set of two clauses:

$$x_1 + x_2 + x_3 \qquad \text{and} \qquad x_4 + \overline{x_1} + x_2.$$

Note, that Max 3SAT is a ***maximization problem***.

Definition 1.2.9. Algorithm **Alg** for a maximization problem achieves an approximation factor α if for all inputs, we have:

$$\frac{\mathbf{Alg}(G)}{\mathrm{Opt}(G)} \geq \alpha.$$

In the following, we present a ***randomized algorithm*** – it is allowed to consult with a source of random numbers in making decisions. A key property we need about random variables, is the linearity of expectation property, which is easy to derive directly from the definition of expectation.

Definition 1.2.10 (***Linearity of expectations***.). Given two random variables X, Y (not necessarily independent, we have that $\mathbb{E}\big[X + Y\big] = \mathbb{E}\big[X\big] + \mathbb{E}\big[Y\big]$.

Theorem 1.2.11. *One can achieve (in expectation) (7/8)-approximation to Max 3SAT in polynomial time. Namely, if the instance has m clauses, then the generated assignment satisfies $(7/8)m$ clauses in* expectation.

Proof: Let $x_1, \ldots, x_n$ be the n variables used in the given instance. The algorithm works by randomly assigning values to $x_1, \ldots, x_n$, independently, and equal probability, to 0 or 1, for each one of the variables.

Let Y_i be the indicator variables which is 1 if (and only if) the ith clause is satisfied by the generated random assignment and 0 otherwise, for $i = 1, \ldots, m$. Formally, we have

$$Y_i = \begin{cases} 1 & C_i \text{ is satisfied by the generated assignment,} \\ 0 & \text{otherwise.} \end{cases}$$

Now, the number of clauses satisfied by the given assignment is $Y = \sum_{i=1}^{m} Y_i$. We claim that $\mathbb{E}[Y] = (7/8)m$, where m is the number of clauses in the input. Indeed, we have

$$\mathbb{E}\big[Y\big] = \mathbb{E}\Big[\sum_{i=1}^{m} Y_i\Big] = \sum_{i=1}^{m} \mathbb{E}\big[Y_i\big]$$

by linearity of expectation. Now, what is the probability that $Y_i = 0$? This is the probability that all three literals appear in the clause C_i are evaluated to FALSE. Since the three literals are instance of three distinct variable, these three events are independent, and as such the probability for this happening is

$$\mathbb{P}\big[Y_i = 0\big] = \frac{1}{2} * \frac{1}{2} * \frac{1}{2} = \frac{1}{8}.$$

(Another way to see this, is to observe that since C_i has exactly three literals, there is only one possible assignment to the three variables appearing in it, such that the clause evaluates to FALSE. Now, there are eight (8) possible assignments to this clause, and thus the probability of picking a FALSE assignment is 1/8.) Thus,

$$\mathbb{P}\big[Y_i = 1\big] = 1 - \mathbb{P}\big[Y_i = 0\big] = \frac{7}{8},$$

and

$$\mathbb{E}\big[Y_i\big] = \mathbb{P}\big[Y_i = 0\big] * 0 + \mathbb{P}\big[Y_i = 1\big] * 1 = \frac{7}{8}.$$

Namely, $\mathbb{E}[\#\text{ of clauses sat}] = \mathbb{E}[Y] = \sum_{i=1}^{m} \mathbb{E}[Y_i] = (7/8)m$. Since the optimal solution satisfies at most m clauses, the claim follows. ∎

Curiously, Theorem 1.2.11 is stronger than what one usually would be able to get for an approximation algorithm. Here, the approximation quality is independent of how well the optimal solution does (the optimal can satisfy at most m clauses, as such we get a (7/8)-approximation. Curiouser and curiouser[①], the algorithm does not even look on the input when generating the random assignment.

Håstad [Hås01a] proved that one can do no better; that is, for any constant $\varepsilon > 0$, one can not approximate 3SAT in polynomial time (unless P = NP) to within a factor of $7/8 + \varepsilon$. It is pretty amazing that a trivial algorithm like the above is essentially optimal.

Remark 1.2.12. For $k \geq 3$, the above implies $1 - 2^{-k}$-approximation algorithm, for k-SAT, as long as the instances are each of length at least k.

[①] "Curiouser and curiouser!" Cried Alice (she was so much surprised, that for the moment she quite forgot how to speak good English). – Alice in wonderland, Lewis Carol

1.3. The Markov and Chebyshev's inequalities

1.3.1. Markov's inequality

We remind the reader that for a random variable X assuming real values, its ***expectation*** is $\mathbb{E}[Y] = \sum_y y \cdot \mathbb{P}[Y = y]$. Similarly, for a function $f(\cdot)$, we have $\mathbb{E}[f(Y)] = \sum_y f(y) \cdot \mathbb{P}[Y = y]$.

Theorem 1.3.1 (Markov's Inequality). *Let Y be a random variable assuming only non-negative values. Then for all $t > 0$, we have*

$$\mathbb{P}\big[Y \geq t\big] \leq \frac{\mathbb{E}\big[Y\big]}{t}.$$

Proof: Indeed,

$$\begin{aligned}\mathbb{E}\big[Y\big] &= \sum_{y \geq t} y\, \mathbb{P}\big[Y = y\big] + \sum_{y < t} y\, \mathbb{P}[Y = y] \geq \sum_{y \geq t} y\, \mathbb{P}\big[Y = y\big] \\ &\geq \sum_{y \geq t} t\, \mathbb{P}\big[Y = y\big] = t\, \mathbb{P}\big[Y \geq t\big].\end{aligned}$$

■

Markov inequality is tight, as the following exercise testifies.

Exercise 1.3.2. For any (integer) $k > 1$, define a random positive variable X_k such that $\mathbb{P}[X_k \geq k\, \mathbb{E}[X_k]] = 1/k$.

1.3.1.1. Another example for expectation

Let $X_i \in \{-1, +1\}$ with probability half for each value, for $i = 1, \ldots, n$ (all picked independently). Let $Y = \sum_i X_i$.

$$\mathbb{E}[Y] = \mathbb{E}\left[\sum_i X_i\right] = \sum_i \mathbb{E}[X_i] = n \cdot 0 = 0.$$

A more interesting quantity is

$$\begin{aligned}\mathbb{E}\big[Y^2\big] &= \mathbb{E}\Big[(\sum_i X_i)^2\Big] = \mathbb{E}\left[\sum_i X_i^2 + 2\sum_{i<j} X_i X_j\right] = \sum_i \mathbb{E}\big[X_i^2\big] + 2\,\mathbb{E}\left[\sum_{i<j} X_i X_j\right] = n + 2\sum_{i<j} \mathbb{E}\big[X_i X_j\big] \\ &= n + 2\sum_{i<j} \mathbb{E}[X_i]\, \mathbb{E}\big[X_j\big] = n.\end{aligned}$$

Lemma 1.3.3. *Let $X_i \in \{-1, +1\}$ with probability half for each value, for $i = 1, \ldots, n$ (all picked independently). We have that $\mathbb{P}\big[|\sum_i X_i| > t\sqrt{n}\big] < 1/t^2$.*

Proof: Let $Y = \sum_i X_i$ and $Z = Y^2$. We have

$$\mathbb{P}\left[\left|\sum_i X_i\right| > t\sqrt{n}\right] = \mathbb{P}\left[\left(\sum_i X_i\right)^2 > t^2 n\right] = \mathbb{P}\big[Y^2 > t^2\, \mathbb{E}\big[Y^2\big]\big] = \mathbb{P}\big[Z > t^2\, \mathbb{E}[Z]\big] \leq 1/t^2,$$

by Markov's inequality. ■

1.3.2. Chebychev's inequality

Theorem 1.3.4 (Chebyshev's inequality). *Let X be a real random variable, with $\mu_X = \mathbb{E}[X]$, and $\sigma_X = \sqrt{\mathbb{V}[X]}$. Then, for any $t > 0$, we have $\mathbb{P}\big[|X - \mu_X| \geq t\sigma_X\big] \leq 1/t^2$.*

Proof: Note that

$$\mathbb{P}\big[|X - \mu_X| \geq t\sigma_X\big] = \mathbb{P}\big[\,(X - \mu_X)^2 \geq t^2\sigma_X^2\big].$$

Set $Y = (X - \mu_X)^2$. Clearly, $\mathbb{E}[.Y] = \sigma_X^2$. Now, apply Markov's inequality to Y. ■

1.4. Quick Sort

Let the input be a set $T = \{t_1, \ldots, t_n\}$ of n items to be sorted. We remind the reader, that the **QuickSort** algorithm randomly pick a pivot element (uniformly), splits the input into two subarrays of all the elements smaller than the pivot, and all the elements larger than the pivot, and then it recurses on these two subarrays (the pivot is not included in these two subproblems). Here we will show that the expected running time of **QuickSort** is $O(n \log n)$.

Definition 1.4.1. For an event $\mathcal{E}$, let X be a random variable which is 1 if $\mathcal{E}$ occurred and 0 otherwise. The random variable X is an ***indicator variable***.

Observation 1.4.2. *For an indicator variable X of an event $\mathcal{E}$, we have*

$$\mathbb{E}\big[X\big] = 0 \cdot \mathbb{P}\big[X = 0\big] + 1 \cdot \mathbb{P}\big[X = 1\big] = \mathbb{P}\big[X = 1\big] = \mathbb{P}\big[\mathcal{E}\big].$$

Let $S_1, \ldots, S_n$ be the elements in their sorted order (i.e., the output order). Let $X_{ij} = 1$ be the indicator variable which is one iff **QuickSort** compares S_i to S_j, and let p_{ij} denote the probability that this happens. Clearly, the number of comparisons performed by the algorithm is $C = \sum_{i<j} X_{ij}$. By linearity of expectations, we have

$$\mathbb{E}\big[C\big] = \mathbb{E}\Big[\sum\nolimits_{i<j} X_{ij}\Big] = \sum_{i<j} \mathbb{E}\big[X_{ij}\big] = \sum_{i<j} p_{ij}.$$

We want to bound p_{ij}, the probability that the S_i is compared to S_j. Consider the last recursive call involving both S_i and S_j. Clearly, the pivot at this step must be one of $S_i, \ldots, S_j$, all equally likely. Indeed, S_i and S_j were separated in the next recursive call.

Observe, that S_i and S_j get compared if and only if pivot is S_i or S_j. Thus, the probability for that is $2/(j - i + 1)$. Indeed,

$$p_{ij} = \mathbb{P}\big[S_i \text{ or } S_j \text{ picked} \,\big|\, \text{picked pivot from } S_i, \ldots, S_j\big] = \frac{2}{j - i + 1}.$$

Thus,

$$\sum_{i=1}^{n} \sum_{j>i} p_{ij} = \sum_{i=1}^{n} \sum_{j>i} 2/(j - i + 1) = \sum_{i=1}^{n} \sum_{k=1}^{n-i+1} \frac{2}{k} \leq 2 \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{1}{k} \leq 2nH_n \leq n + 2n \ln n,$$

where H_n is the ***harmonic number***[2] $H_n = \sum_{i=1}^{n} 1/i$, We thus proved the following result.

[2] Using integration to bound summation, we have $H_n \leq 1 + \int_{x=1}^{n} \frac{1}{x} dx \leq 1 + \ln n$. Similarly, $H_n \geq \int_{x=1}^{n} \frac{1}{x} dx = \ln n$.

Lemma 1.4.3. **QuickSort** *performs in expectation at most* $n + 2n \ln n$ *comparisons, when sorting* n *elements.*

Note, that this holds for all inputs. No assumption on the input is made. Similar bounds holds not only in expectation, but also with high probability.

This raises the question, of how does the algorithm pick a random element? We assume we have access to a random source that can get us number between 1 and n uniformly.

Note, that the algorithm always works, but it might take quadratic time in the worst case.

Remark 1.4.4 (Wait, wait, wait). Let us do the key argument in the above more slowly, and more carefully. Imagine, that before running **QuickSort** we choose for every element a random priority, which is a real number in the range $[0, 1]$. Now, we reimplement **QuickSort** such that it always pick the element with the lowest random priority (in the given subproblem) to be the pivot. One can verify that this variant and the standard implementation have the same running time. Now, a_i gets compares to a_j if and only if all the elements $a_{i+1}, \ldots, a_{j-1}$ have random priority larger than both the random priority of a_i and the random priority of a_j. But the probability that one of two elements would have the lowest random-priority out of $j - i + 1$ elements is $2 * 1/(j - i + 1)$, as claimed.

Chapter 2

Quick Sort with High Probability

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
December 10, 2019

2.1. Conditional expectation

Definition 2.1.1 (Conditional Probability). The ***conditional probability*** of X given Y, is the probability that $X = x$ given that $Y = y$. We denote this quantity by $\mathbb{P}[X = x \mid Y = y]$.

For two random variables X and Y, let $\mathbb{E}[X \mid Y]$ denote the expected value of X, if the value of Y is specified. Formally, we have

$$\mathbb{E}[X \mid Y = y] = \sum_{x \in \Omega} x \, \mathbb{P}[X = x \mid Y = y] \, .$$

The expression $\mathbb{E}[X \mid Y]$, which is a shorthand for $\mathbb{E}[X \mid Y = y]$, is the ***conditional expectation*** of X given Y. In reality, it is a function from the value of y, to the average value of X. As such, one can think of conditional expectation as a function $f(y) = \mathbb{E}[X \mid Y = y]$.

Lemma 2.1.2. *For any two random variables X and Y, we have $\mathbb{E}\big[\mathbb{E}[X \mid Y]\big] = \mathbb{E}\big[X\big]$.*

Proof: $\mathbb{E}\big[\mathbb{E}[X \mid Y]\big] = \mathbb{E}_Y\big[\mathbb{E}\big[X \,\big|\, Y = y\big]\big] = \sum_y \mathbb{P}[Y = y] \, \mathbb{E}[X \mid Y = y]$

$$= \sum_y \mathbb{P}\big[Y = y\big] \sum_x x \, \mathbb{P}[X = x \mid Y = y] = \sum_y \mathbb{P}\big[Y = y\big] \frac{\sum_x x \, \mathbb{P}[X = x \cap Y = y]}{\mathbb{P}[Y = y]}$$
$$= \sum_y \sum_x x \, \mathbb{P}\big[X = x \cap Y = y\big] = \sum_x x \sum_y \mathbb{P}\big[X = x \cap Y = y\big] = \sum_x x \, \mathbb{P}\big[X = x\big] = \mathbb{E}\big[X\big] \, . \qquad \blacksquare$$

Lemma 2.1.3. *For any two random variables X and Y, we have $\mathbb{E}\big[Y \cdot \mathbb{E}[X \mid Y]\big] = \mathbb{E}\big[XY\big]$.*

Proof: We have that $\mathbb{E}\big[Y \cdot \mathbb{E}\big[X \,\big|\, Y\big]\big] = \sum_y \mathbb{P}[Y = y] \cdot y \cdot \mathbb{E}\big[X \,\big|\, Y = y\big]$

$$= \sum_y \mathbb{P}[Y = y] \cdot y \cdot \frac{\sum_x x \, \mathbb{P}[X = x \cap Y = y]}{\mathbb{P}[Y = y]} = \sum_x \sum_y xy \cdot \mathbb{P}\big[X = x \cap Y = y\big] = \mathbb{E}\big[XY\big] \, . \qquad \blacksquare$$

2.2. QuickSort runs in $O(n \log n)$ time with high probability

Consider a set T of the n items to be sorted, and consider a specific element $t \in T$. Let X_i be the size of the input in the ith level of recursion that contains t. We know that $X_0 = n$, and

$$\mathbb{E}\left[X_i \,\middle|\, X_{i-1}\right] \leq \frac{1}{2}\frac{3}{4}X_{i-1} + \frac{1}{2}X_{i-1} \leq \frac{7}{8}X_{i-1}.$$

Indeed, with probability $1/2$ the pivot is the middle of the subproblem; that is, its rank is between $X_{i-1}/4$ and $(3/4)X_{i-1}$ (and then the subproblem has size $\leq X_{i-1}(3/4)$), and with probability $1/2$ the subproblem might has not shrank significantly (i.e., we pretend it did not shrink at all).

Now, observe that for any two random variables we have that $\mathbb{E}[X] = \mathbb{E}_y[\mathbb{E}[X \,|\, Y = y]]$, see Lemma 2.1.2_{p19}.. As such, we have that

$$\mathbb{E}[X_i] = \mathop{\mathbb{E}}_{y}\left[\mathbb{E}\left[X_i \,\middle|\, X_{i-1} = y\right]\right] \leq \mathop{\mathbb{E}}_{X_{i-1}=y}\left[\frac{7}{8}y\right] = \frac{7}{8}\mathbb{E}[X_{i-1}] \leq \left(\frac{7}{8}\right)^i \mathbb{E}[X_0] = \left(\frac{7}{8}\right)^i n.$$

In particular, consider $M = 8\log_{8/7} n$. We have that

$$\mu = \mathbb{E}[X_M] \leq \left(\frac{7}{8}\right)^M n \leq \frac{1}{n^8}n = \frac{1}{n^7}.$$

Of course, t participates in more than M recursive calls, if and only if $X_M \geq 1$. However, by Markov's inequality (Theorem 1.3.1), we have that

$$\mathbb{P}\left[\begin{array}{c}\text{element } t \text{ participates}\\ \text{in more than } M \text{ recursive calls}\end{array}\right] \leq \mathbb{P}[X_M \geq 1] \leq \frac{\mathbb{E}[X_M]}{1} \leq \frac{1}{n^7},$$

as desired. That is, we proved that the probability that any element of the input T participates in more than M recursive calls is at most $n(1/n^7) \leq 1/n^6$.

Theorem 2.2.1. *For n elements, QuickSort runs in $O(n \log n)$ time, with high probability.*

2.3. Treaps

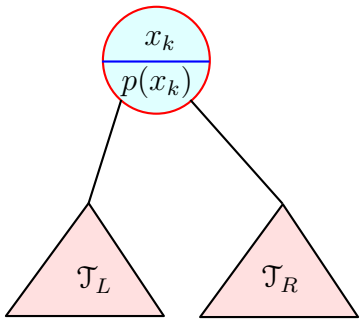
Anybody that ever implemented a balanced binary tree, knows that it can be very painful. A natural question, is whether we can use randomization to get a simpler data-structure with good performance.

2.3.1. Construction

The key observation is that many of data-structures that offer good performance for balanced binary search trees, do so by storing additional information to help in how to balance the tree. As such, the key Idea is that for every element x inserted into the data-structure, randomly choose a priority $p(x)$; that is, $p(x)$ is chosen uniformly and randomly in the range $[0, 1]$.

So, for the set of elements $X = \{x_1, \ldots, x_n\}$, with (random) priorities $p(x_1), \ldots, p(x_n)$, our purpose is to build a binary tree which is "balanced". So, let us pick the element x_k with the lowest priority in X, and make it the root of the tree. Now, we partition X in the natural way:

(A) L: set of all the numbers smaller than x_k in X, and



(B) R: set of all the numbers larger than x_k in X.

We can now build recursively the trees for L and R, and let denote them by T_L and T_R. We build the natural tree, by creating a node for x_k, having T_L its left child, and T_R as its right child.

We call the resulting tree a ***treap***. As it is a tree over the elements, and a heap over the priorities; that is, TREAP = TREE + HEAP.

Lemma 2.3.1. *Given n elements, the expected depth of a treap T defined over those elements is $O(\log(n))$. Furthermore, this holds with high probability; namely, the probability that the depth of the treap would exceed $c \log n$ is smaller than $\delta = n^{-d}$, where d is an arbitrary constant, and c is a constant that depends on d.*[1]

Furthermore, the probability that T has depth larger than $ct\log(n)$, for any $t \geq 1$, is smaller than n^{-dt}.

Proof: Observe, that every element has equal probability to be in the root of the treap. Thus, the structure of a treap, is identical to the recursive tree of **QuickSort**. Indeed, imagine that instead of picking the pivot uniformly at random, we instead pick the pivot to be the element with the lowest (random) priority. Clearly, these two ways of choosing pivots are equivalent. As such, the claim follows immediately from our analysis of the depth of the recursion tree of **QuickSort**, see Theorem 2.2.1. ■

2.3.2. Operations

The following innocent observation is going to be the key insight in implementing operations on treaps:

Observation 2.3.2. *Given n distinct elements, and their (distinct) priorities, the treap storing them is uniquely defined.*

2.3.2.1. Insertion

Given an element x to be inserted into an existing treap T, insert it in the usual way into T (i.e., treat it a regular search binary tree). This takes $O(\text{height}(T))$. Now, x is a leaf in the treap. Set x priority $p(x)$ to some random number $[0, 1]$. Now, while the new tree is a valid search tree, it is not necessarily still a valid treap, as x's priority might be smaller than its parent. So, we need to fix the tree around x, so that the priority property holds.

[1] That is, if we want to decrease the probability of failure, that is δ, we need to increase c.

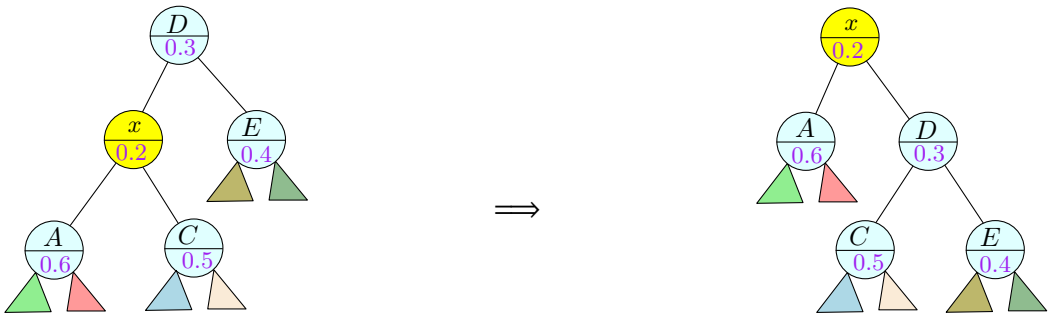


Figure 2.1: **RotateRight**: Rotate right in action.

```
RotateUp(x)
    y ← parent(x)
    while p(y) > p(x) do
        if y.left_child = x then
            RotateRight(y)
        else
            RotateLeft(y)
        y ← parent(x)
```

We call **RotateUp**(x) to do so. Specifically, if x parent is y, and $p(x) < p(y)$, we will rotate x up so that it becomes the parent of y. We repeatedly do it till x has a larger priority than its parent. The rotation operation takes constant time and plays around with priorities, and importantly, it preserves the binary search tree order. Here is a rotate right operation **RotateRight**(D):

RotateLeft is the same tree rewriting operation done in the other direction.

In the end of this process, both the ordering property and the priority property holds. That is, we have a valid treap that includes all the old elements, and the new element. By Observation 2.3.2, since the treap is uniquely defined, we have updated the treap correctly. Since every time we do a rotation the distance of x from the root decrease by one, it follows that insertions takes $O(\text{height}(T))$.

2.3.2.2. Deletion

Deletion is just an insertion done in reverse. Specifically, to delete an element x from a treap T, set its priority to $+\infty$, and rotate it down it becomes a leaf. The only tricky observation is that you should rotate always so that the child with the lower priority becomes the new parent. Once x becomes a leaf deleting it is trivial - just set the pointer pointing to it in the tree to null.

2.3.2.3. Split

Given an element x stored in a treap T, we would like to split T into two treaps – one treap $T_{\leq}$ for all the elements smaller or equal to x, and the other treap $T_{>}$ for all the elements larger than x. To this end, we set x priority to $-\infty$, fix the priorities by rotating x up so it becomes the root of the treap. The right child of x is the treap $T_{>}$, and we disconnect it from T by setting x right child pointer to null. Next, we restore x to its real priority, and rotate it down to its natural location. The resulting treap is $T_{\leq}$. This again takes time that is proportional to the depth of the treap.

2.3.2.4. Meld

Given two treaps T_L and T_R such that all the elements in T_L are smaller than all the elements in T_R, we would like to merge them into a single treap. Find the largest element x stored in T_L (this is just the element stored in the path going only right from the root of the tree). Set x priority to $-\infty$, and rotate it up the treap so that it becomes the root. Now, x being the largest element in T_L has no right child. Attach T_R as the right child of x. Now, restore x priority to its original priority, and rotate it back so the priorities properties hold.

2.3.3. Summery

Theorem 2.3.3. *Let T be a treap, initialized to an empty treap, and undergoing a sequence of $m = n^c$ insertions, where c is some constant. The probability that the depth of the treap in any point in time would exceed $d \log n$ is $\leq 1/n^f$, where d is an arbitrary constant, and f is a constant that depends only c and d.*

In particular, a treap can handle insertion/deletion in $O(\log n)$ time with high probability.

Proof: Since the first part of the theorem implies that with high probability all these treaps have logarithmic depth, then this implies that all operations takes logarithmic time, as an operation on a treap takes at most the depth of the treap.

As for the first part, let $T_1, \ldots, T_m$ be the sequence of treaps, where T_i is the treap after the ith operation. Similarly, let X_i be the set of elements stored in T_i. By Lemma 2.3.1, the probability that T_i has large depth is tiny. Specifically, we have that

$$\alpha_i = \mathbb{P}[\operatorname{depth}(T_i) > tc' \log n^c] = \mathbb{P}\left[\operatorname{depth}(T_i) > c't\left(\frac{\log n^c}{\log |T_i|}\right) \cdot \log |T_i|\right] \leq \frac{1}{n^{t \cdot c}},$$

as a tedious and boring but straightforward calculation shows. Picking t to be sufficiently large, we have that the probability that the ith treap is too deep is smaller than $1/n^{f+c}$. By the union bound, since there are n^c treaps in this sequence of operations, it follows that the probability of any of these treaps to be too deep is at most $1/n^f$, as desired. ■

2.4. Extra: Sorting Nuts and Bolts

Problem 2.4.1 (Sorting Nuts and Bolts). You are given a set of n nuts and n bolts. Every nut have a matching bolt, and all the n pairs of nuts and bolts have different sizes. Unfortunately, you get the nuts and bolts separated from each other and you have to match the nuts to the bolts. Furthermore, given a nut and a bolt, all you can do is to try and match one bolt against a nut (i.e., you can not compare two nuts to each other, or two bolts to each other).

When comparing a nut to a bolt, either they match, or one is smaller than other (and you known the relationship after the comparison).

How to match the n nuts to the n bolts quickly? Namely, while performing a small number of comparisons.

The naive algorithm is of course to compare each nut to each bolt, and match them together. This would require a quadratic number of comparisons. Another option is to sort the nuts by size, and the bolts by size and then "merge" the two ordered sets, matching them by size. The only problem is that we can not sorts only the nuts, or only the bolts, since we can not compare them to each other. Indeed, we sort the two sets simultaneously, by simulating **QuickSort**. The resulting algorithm is depicted on the right.

MatchNuts&Bolts (N: nuts, B: bolts)
  Pick a random nut n_{pivot} from N
  Find its matching bolt b_{pivot} in B
  $B_L \leftarrow$ All bolts in B smaller than n_{pivot}
  $N_L \leftarrow$ All nuts in N smaller than b_{pivot}
  $B_R \leftarrow$ All bolts in B larger than n_{pivot}
  $N_R \leftarrow$ All nuts in N larger than b_{pivot}
  MatchNuts&Bolts(N_R,B_R)
  MatchNuts&Bolts(N_L,B_L)

2.4.1. Running time analysis

Definition 2.4.2. Let $\mathcal{RT}$ denote the random variable which is the running time of the algorithm. Note, that the running time is a random variable as it might be different between different executions on the *same input*.

Definition 2.4.3. For a randomized algorithm, we can speak about the expected running time. Namely, we are interested in bounding the quantity $\mathbb{E}[\mathcal{RT}]$ for the worst input.

Definition 2.4.4. The ***expected running-time*** of a randomized algorithm for input of size n is

$$T(n) = \max_{U \text{ is an input of size } n} \mathbb{E}[\mathcal{RT}(U)],$$

where $\mathcal{RT}(U)$ is the running time of the algorithm for the input U.

Definition 2.4.5. The ***rank*** of an element x in a set S, denoted by $\operatorname{rank}(x)$, is the number of elements in S of size smaller or equal to x. Namely, it is the location of x in the sorted list of the elements of S.

Theorem 2.4.6. *The expected running time of* **MatchNuts&Bolts** *(and thus also of* **QuickSort***) is $T(n) = O(n \log n)$, where n is the number of nuts and bolts. The worst case running time of this algorithm is $O(n^2)$.*

Proof: Clearly, we have that $\mathbb{P}\left[\operatorname{rank}(n_{pivot}) = k\right] = \frac{1}{n}$. Furthermore, if the rank of the pivot is k then

$$\begin{aligned} T(n) &= \mathop{\mathbb{E}}_{k=\operatorname{rank}(n_{pivot})}[O(n) + T(k-1) + T(n-k)] = O(n) + \mathop{\mathbb{E}}_{k}[T(k-1) + T(n-k)] \\ &= T(n) = O(n) + \sum_{k=1}^{n} \mathbb{P}[Rank(Pivot) = k] * (T(k-1) + T(n-k)) \\ &= O(n) + \sum_{k=1}^{n} \frac{1}{n} \cdot (T(k-1) + T(n-k)), \end{aligned}$$

by the definition of expectation. It is not easy to verify that the solution to the recurrence $T(n) = O(n) + \sum_{k=1}^{n} \frac{1}{n} \cdot (T(k-1) + T(n-k))$ is $O(n \log n)$. ■

2.5. Bibliographical Notes

Treaps were invented by Siedel and Aragon [SA96]. Experimental evidence suggests that Treaps performs reasonably well in practice, despite their simplicity, see for example the comparison carried out by Cho and Sahni [CS00]. Implementations of treaps are readily available. An old implementation I wrote in `C` is available here: `http://valis.cs.uiuc.edu/blog/?p=6060`.

Chapter 3

On k-wise independence

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
December 10, 2019

3.1. Pairwise independence

3.1.1. Pairwise independence

Definition 3.1.1. A set of random variables $X_1, \ldots, X_n$ is ***pairwise independent***, if for any pair of values α, β, and any two indices i, j, we have that

$$\mathbb{P}\big[X_i = \alpha \text{ and } Y_j = \beta\big] = \mathbb{P}[X_i = \alpha]\, \mathbb{P}\big[Y_j = \beta\big].$$

Namely, the variables are independent if you look at pairs of variables. Compare this to the much stronger property of independence.

Definition 3.1.2. A set of random variables $X_1, \ldots, X_n$ is ***independent***, if for any t, and any t values $\alpha_1, \ldots, \alpha_t$, and any t indices $i_1, \ldots, i_t$, we have that

$$\mathbb{P}\big[X_{i_1} = \alpha_1, X_{i_2} = \alpha_2, \ldots, \text{ and } Y_{i_t} = \alpha_{i_t}\big] = \prod_{j=1}^{t} \mathbb{P}\big[X_{i_j} = \alpha_j\big].$$

3.1.2. A pairwise independent set of bits

Let n be a number which is a power of two. As such, $t = \log_2 n = \lg n$ is an integer. Let $X_0, \ldots, X_{t-1}$ be truly independent random bits, each one of them is 1 with probability 1/2.

For a non-negative integer number x, let $\mathrm{bit}(x, j) \in \{0, 1\}$ be the jth bit in the binary representation of x. That is, we have $x = \sum_j \mathrm{bit}(x, j) 2^j$.

For an index $i = 1, \ldots, 2^t - 1$, we define $Y_i = \bigotimes_{j:\mathrm{bit}(i,j)=1} X_j$, where $\otimes$ is the *xor* operator.

Lemma 3.1.3. *The random variables $Y_1, Y_2, \ldots, Y_{n-1}$ are pairwise independent.*

Proof: Consider two distinct indices i, i', and two arbitrary values v, v'. We need to prove that

$$\mathbb{P}[Y_i = v \text{ and } Y_{i'} = v'] = \mathbb{P}[Y_i = v]\, \mathbb{P}[Y_{i'} = v'] = \frac{1}{4}.$$

Here, we used that $\mathbb{P}[Y_i = 1] = \mathbb{P}[Y_i = 0] = 1/2$. To see this, let α be an index such that $\mathrm{bit}(i, \alpha) = 1$, and observe that this follows readily if pick all the true random variables $X_0, \ldots, X_{t-1}$ in such an order such that X_α is the last one to be set.

Let $B = \{j \mid \mathrm{bit}(i,j) = 1\}$ and $B' = \{j \mid \mathrm{bit}(i',j) = 1\}$. If there is an index $\beta \in B \setminus B'$, then we have

$$\mathbb{P}[Y_i = v \mid Y_{i'} = v'] = \mathbb{P}\left[\bigotimes_{j:\mathrm{bit}(i,j)=1} X_j = v \;\middle|\; Y_{i'} = v'\right] = \mathbb{P}\left[X_\beta \otimes \bigotimes_{j:\mathrm{bit}(i,j)=1} X_j = v \;\middle|\; Y_{i'} = v'\right]$$
$$= \mathbb{P}\left[X_\beta = \left(v \otimes \bigotimes_{j:\mathrm{bit}(i,j)=1} X_j\right) \;\middle|\; Y_{i'} = v'\right] = \frac{1}{2}.$$

This implies that $\mathbb{P}[Y_i = v \text{ and } Y_{i'} = v'] = \mathbb{P}[Y_i = v \mid Y_{i'} = v']\,\mathbb{P}[Y_{i'} = v'] = (1/2)(1/2) = 1/4$, as claimed.

A similar argument implies that if there is an index $\beta \in B' \setminus B$, then $\mathbb{P}[Y_{i'} = v' \mid Y_i = v] = 1/2$, which implies the claim in this case.

Since $i \neq i'$, one of the two scenarios must happen, implying the claim. ∎

3.1.3. An application: Max cut

Given a graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ with n vertices and m edges, consider the problem of computing the max-cut. That is, computing the set of vertices S, such that the cut

$$(S, \overline{S}) = (S, \mathsf{V} \setminus S) = \{uv \in \mathsf{E} \mid u \in S, v \in \mathsf{V} \setminus S\}.$$

is of maximum cardinality.

To this end, let $Y_1, \ldots, Y_n$ be the pairwise independent bits of Section 3.1.2. Here, let S be the set of all vertices $v_i \in \mathsf{V}$, such that $Y_i = 1$. Let Z_{uv} be an indicator variable for the event that the edge $uv \in \mathsf{E}$ is in the cut $(S, \overline{S})$.

We have that

$$\mathbb{E}\left[\left|(S, \overline{S})\right|\right] = \mathbb{E}\left[\sum_{uv \in \mathsf{E}} Z_{uv}\right] = \sum_{uv \in \mathsf{E}} \mathbb{E}[Z_{uv}] = \sum_{uv \in \mathsf{E}} \mathbb{P}[Y_u \neq Y_v] = 1/2,$$

using linearity of expectation and independence.

Lemma 3.1.4. *Given a graph G with n vertices and m edges, say stored in a read only memory, one can compute a max-cut of G, and the edges in it, using $O(\log n)$ random bits, and $O(\log n)$ RAM bits. Furthermore, the expected size of the cut is $\geq m/2$.*

Proof: The algorithm description is above. The pairwise independence is described above requires $O(\log n)$ random bits, which needs to be stored. Otherwise, all we need is to scan the edges of the graph, and for each one to decide if it is or not in the graph. Clearly, this can be done using $O(\log n)$ RAM bits. ∎

Compare this to the natural randomized algorithm of computing a random subset S. This would require using n random bits, and n bits of space to store it.

Max cut in the streaming model. Imagine that the edges of the graph are given to you via streaming: You are told the number of vertices in advance, but then edges arrive one by one. The above enables you to compute the cut in a streaming fashion using $O(\log n)$ bits. Alternatively, you can output the edges in a streaming fashion.

Another way of thinking about it, is that given a set $S = \{s_1, \ldots, s_n\}$ of n elements, we can use the above to select a random sample where every element is selected with probability half, and the samples are pairwise independent. The kicker is that to specify the sample, or decide if an element is in the sample, we can do it using $O(\log n)$ bits. This is a huge save compared to the regular n bits required to maintain to remember the sample.

It is clear however that we want a stronger concept – where things are k-wise independent.

3.2. On k-wise independence

3.2.1. Definition

Definition 3.2.1. A set of variables $X_1, \ldots, X_n$ are ***k-wise independent*** if for any set $I = \{i_1, i_2, \ldots, i_t\}$ of indices, for $t \leq k$, and any set of values $v_1, \ldots, v_t$, we have that

$$\mathbb{P}\big[X_{i_1} = v_1 \text{ and } X_{i_2} = v_2 \text{ and } \cdots \text{ and } X_{i_t} = v_t\big] = \prod_{j=1}^{t} \mathbb{P}\big[X_{i_j} = v_j\big].$$

Observe, that verifying the above property needs to be done only for $t = k$.

3.2.2. On working modulo prime

Definition 3.2.2. For a number p, let $\mathbb{Z}_n = \big\{0, \ldots, n-1\big\}$.

For two integer numbers x and y, the ***quotient*** of x/y is $x \operatorname{div} y = \lfloor x/y \rfloor$. The ***remainder*** of x/y is $x \bmod y = x - y\lfloor x/y \rfloor$. If the $x \bmod y = 0$, than y ***divides*** x, denoted by $y \mid x$. We use $\alpha \equiv \beta \pmod p$ or $\alpha \equiv_p \beta$ to denote that α and β are ***congruent modulo*** p; that is $\alpha \bmod p = \beta \bmod p$ – equivalently, $p \mid (\alpha - \beta)$.

Lemma 3.2.3. *Let p be a prime number.*

(A) For any $\alpha, \beta \in \{1, \ldots, p-1\}$, we have that $\alpha\beta \not\equiv 0 \pmod p$.

(B) For any $\alpha, \beta, i \in \{1, \ldots, p-1\}$, such that $\alpha \neq \beta$, we have that $\alpha i \not\equiv \beta i \pmod p$.

(C) For any $x \in \{1, \ldots, p-1\}$ there exists a unique y such that $xy \equiv 1 \pmod p$. The number y is the **inverse** *of x, and is denoted by x^{-1} or $1/x$.*

Proof: (A) If $\alpha\beta \equiv 0 \pmod p$, then p must divide $\alpha\beta$, as it divides 0. But α, β are smaller than p, and p is prime. This implies that either $p \mid \alpha$ or $p \mid \beta$, which is impossible.

(B) Assume that $\alpha > \beta$. Furthermore, for the sake of contradiction, assume that $\alpha i \equiv \beta i \pmod p$. But then, $(\alpha - \beta)i \equiv 0 \pmod p$, which is impossible, by (A).

(C) For any $\alpha \in \{1, \ldots, p-1\}$, consider the set $L_\alpha = \{\alpha * 1 \bmod p, \alpha * 2 \bmod p, \ldots, \alpha * (p-1) \bmod p\}$. By (A), zero is not in L_α, and by (B), L_α must contain $p-1$ distinct values. It follows that $L_\alpha = \{1, 2, \ldots, p-1\}$. As such, there exists exactly one number $y \in \{1, \ldots, p-1\}$, such that $\alpha y \equiv 1 \pmod p$. ■

Lemma 3.2.4. *Consider a prime p, and any numbers $x, y \in \mathbb{Z}_p$. If $x \neq y$ then, for any $a, b \in \mathbb{Z}_p$, such that $a \neq 0$, we have $ax + b \not\equiv ay + b \pmod p$.*

Proof: Assume $y > x$ (the other case is handled similarly). If $ax + b \equiv ay + b \pmod p$ then $a(x - y) \pmod p = 0$ and $a \neq 0$ and $(x - y) \neq 0$. However, a and $x - y$ cannot divide p since p is prime and $a < p$ and $0 < x - y < p$. ■

Lemma 3.2.5. *Consider a prime p, and any numbers $x, y \in \mathbb{Z}_p$. If $x \neq y$ then, for each pair of numbers $r, s \in \mathbb{Z}_p = \{0, 1, \ldots, p-1\}$, such that $r \neq s$, there is exactly one unique choice of numbers $a, b \in \mathbb{Z}_p$ such that $ax + b \pmod p = r$ and $ay + b \pmod p = s$.*

Proof: Solve the system of equations

$$ax + b \equiv r \pmod p \qquad \text{and} \qquad ay + b \equiv s \pmod p.$$

We get $a = \frac{r-s}{x-y} \pmod p$ and $b = r - ax \pmod p$. ■

3.2.3. Construction of k-wise independence variables

3.2.4. Construction

Consider the following matrix, aka the ***Vandermonde matrix***, defined by n variables:

$$V = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{n-1} \\ 1 & x_3 & x_3^2 & \dots & x_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^{n-1} \end{bmatrix}.$$

Claim 3.2.6. $\det(V) = \prod_{1 \le i < j \le n} (x_j - x_i)$.

Proof: One can prove this in several ways, and we include a proof via properties of polynomials. The determinant $\det(V)$ is a polynomial in the variables $x_1, x_2, \dots, x_n$. Formally, let Π be the set of all permutations of $[\![n]\!] = \{1, \dots, n\}$. For a permutation $\pi \in \Pi$, let $\mathrm{sign}(\pi) \in \{-1, +1\}$ denote the sign of this permutation. We have that

$$f(x_1, x_2, \dots, x_n) = \det(V) = \sum_{\pi \in \Pi} \mathrm{sign}(\pi) x_i^{\pi(i)}.$$

Every monomial in this polynomial has total degree $\sum_{i=1}^{n} \pi(i) = 1 + 2 + \cdots + n = n(n-1)/2$. Observe, that if we replace x_j by x_i, then we have $f(x_1, \dots, x_i, \dots, x_{j-1}, x_i, x_{j+1}, \dots, x_n)$ is the determinant of a matrix with two identical rows, and such a matrix has a zero determinate. Namely, the polynomial f is zero if $x_i = x_j$. This implies that $x_j - x_i$ divides f. We conclude that the polynomial $g \equiv \prod_{1 \le i < j \le n} (x_j - x_i)$ divides f. Namely, we can write $f = g * h$, where h is some polynomial.

Consider the monomial $x_2 x_3^2 \cdots x_n^{n-1}$. It appears in f with coefficient 1. Similarly, it generated in g by selecting the first term in each sub-polynomial, that is $\prod_{1 \le i < j \le n} \Big(x_j - x_i \Big)$. It is to verify that this is the only time this monomial appears in g. This implies that $h = 1$. We conclude that $f = g$, as claimed. ■

Lemma 3.2.7. *For a vector $\mathsf{b} = (b_0, \dots, b_{k-1}) \in \mathbb{Z}_p^k$, consider the associated polynomial $f(x, \mathsf{b}) = \sum_{i=0}^{k-1} b_i x^i \bmod p$. For any k distinct values $\alpha_1, \dots, \alpha_k \in \mathbb{Z}_p$, and k values $v_1, \dots, v_k \in \mathbb{Z}_p$, then there is a unique choice of b, such that $f(\alpha_i) = v_i \bmod p$, for $i = 1, \dots, k$.*

Proof: Let $\alpha_i = \left(1, \alpha_i, \alpha_i^2, \cdots, \alpha_i^{k-1}\right)$. We have that $f(\alpha_i, \mathsf{b}) = \langle \alpha_i, \mathsf{b} \rangle \bmod p$. This translates into the linear system

$$\begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_k \end{pmatrix} \mathsf{b}^T = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \end{pmatrix} \iff \mathsf{M}\,\mathsf{b}^T = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \end{pmatrix} \qquad \text{where} \qquad \mathsf{M} = \begin{bmatrix} 1 & \alpha_1 & \alpha_1^2 & \dots & \alpha_1^{n-1} \\ 1 & \alpha_2 & \alpha_2^2 & \dots & \alpha_2^{n-1} \\ 1 & \alpha_3 & \alpha_3^2 & \dots & \alpha_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_n & \alpha_n^2 & \dots & \alpha_n^{n-1} \end{bmatrix}.$$

The matrix M is the Vandermonde matrix, and by the above it is invertible. We thus get there exists a unique solution to this system of linear equations (modulo p). ■

The construction. So, let us pick independently and uniformly k values $b_0.b_1, \ldots, b_{k-1} \in \mathbb{Z}_p$, let $\mathsf{b} = (b_0, b_1, \ldots, b_{k-1})$. $g(x) = \sum_{i=0}^{k-1} b_i x^i \bmod p$, and consider the random variables

$$Y_i = g(i), \qquad \forall i \in \mathbb{Z}_p.$$

Lemma 3.2.8. *The variables $Y_0, \ldots, Y_{p-1}$ are uniformly distributed and k-wise independent.*

Proof: The uniform distribution for each Y_i follows readily by picking b_0 last, and observing that each such choice corresponds to a different value of Y_i.

As for the k-independence, observe that for any set $I = \{i_1, i_2, \ldots, i_k\}$ of indices, for $t \le k$, and any set of values $v_1, \ldots, v_k \in \mathbb{Z}_p$, we have that the event

$$Y_{i_1} = v_1 \text{ and } Y_{i_2} = v_2 \text{ and } \cdots \text{ and } Y_{i_k} = v_k$$

happens only for a unique choice of b, by Lemma 3.2.7. But there are p^k such choices. We conclude that the probability of the above event is $1/p^k = \prod_{j=1}^{k} \mathbb{P}\left[Y_{i_j} = v_j\right]$, as desired. ∎

We summarize the result for later use.

Theorem 3.2.9. *let p be a prime number, and pick independently and uniformly k values $b_0.b_1, \ldots, b_{k-1} \in \mathbb{Z}_p$, and let $g(x) = \sum_{i=0}^{k-1} b_i x^i \bmod p$. Then the random variables*

$$Y_0 = g(0), \ldots, Y_{p-1} = g(p-1).$$

are uniformly distributed in $\mathbb{Z}_p$ and are k-wise independent.

3.2.5. Applications of k-wide independent variables

Lemma 3.2.10. *If $X_1, \ldots, X_k$ are k-wise independent, then $\mathbb{E}[X_1 \cdots X_k] = \mathbb{E}[X_1] \cdots \mathbb{E}[X_k]$.*

3.3. Higher moment inequalities

The following is the higher moment variant of Chebychev inequality.

Lemma 3.3.1. *For a random variable X, we have that $\mathbb{P}\left[|X - \mathbb{E}[X]| \ge t\mathbb{E}\left[|X - \mathbb{E}[X]|^k\right]^{1/k}\right] \le \frac{1}{t^k}$*

Proof: Setting $Z = |X - \mathbb{E}[X]|^k$, and raising the inequality by a power of k, we have

$$\mathbb{P}\left[|X - \mathbb{E}[X]| \ge t\mathbb{E}\left[|X - \mathbb{E}[X]|^k\right]^{1/k}\right] = \mathbb{P}\left[Z^{1/k} \ge t\,\mathbb{E}[Z]^{1/k}\right] = \mathbb{P}\left[Z \ge t^k\,\mathbb{E}[Z]\right] \le \frac{1}{t^k},$$

by Markov's inequality. ∎

The problem is that computing (or even bounding) the kth moment $M_k(X) = \mathbb{E}\left[|X - \mathbb{E}[X]\,|^k\right]$ is usually not easy. Let us do it for one interesting example.

Lemma 3.3.2. *Consider k be an even integer and let $X_1, \ldots, X_n$ be n random independent variables such that $\mathbb{P}[X_i = -1] = \mathbb{P}[X_i = +1] = 1/2$. Let $X = \sum_{i=1}^{n} X_i$. Then, we have*

$$\mathbb{P}\left[|X| \ge \frac{tk}{2}\sqrt{n}\right] \le \frac{1}{t^k}.$$

Proof: Observe that $\mathbb{E}[X] = n\,\mathbb{E}[X_1] = 0$. We are interested in computing

$$M_k(X) = \mathbb{E}\left[X^k\right] = \mathbb{E}\left[\left(\sum_i X_i\right)^k\right] = \mathbb{E}\left[\sum_{i_1=1}^{n} \cdots \sum_{i_k=1}^{n} X_{i_1} X_{i_2} \cdots X_{i_k}\right] = \sum_{i_1=1}^{n} \cdots \sum_{i_k=1}^{n} \mathbb{E}\left[X_{i_1} X_{i_2} \cdots X_{i_k}\right] \tag{3.1}$$

Consider a term in the above summation, where one of the indices (say i_1) has a unique value among $i_1, i_2, \ldots, i_k$. By independence, we have

$$\mathbb{E}\left[X_{i_1} X_{i_2} \cdots X_{i_k}\right] = \mathbb{E}\left[X_{i_1}\right] \mathbb{E}\left[X_{i_2} \cdots X_{i_k}\right] = 0,$$

since $\mathbb{E}\left[X_{i_1}\right] = 0$. As such, in the above all terms that have a unique index disappear. A term that does not disappear is going to be of the form

$$\mathbb{E}\left[X_{i_1}^{\alpha_1} X_{i_2}^{\alpha_2} \ldots X_{i_\ell}^{\alpha_\ell}\right] = \mathbb{E}\left[X_{i_1}^{\alpha_1}\right] \mathbb{E}\left[X_{i_2}^{\alpha_2}\right] \ldots \mathbb{E}\left[X_{i_\ell}^{\alpha_\ell}\right]$$

where $\alpha_i \geq 2$, and $\sum_i \alpha_i = k$. Observe that

$$\mathbb{E}\left[X_1^t\right] = \begin{cases} 0 & t \text{ is odd} \\ 1 & t \text{ is even.} \end{cases}$$

As such, all the terms in the summation of Eq. (3.1) that have value that is not zero, have value one. These terms corresponds to tuples $T = (i_1, i_2, \ldots, i_k)$, such that the set of values $I(T) = \{i_1, \ldots, i_k\}$ has at most $k/2$ values, and furthermore, each such value appears an even number of times in T. We conclude that the total number of such tuples is at most

$$n^{k/2}(k/2)^k.$$

Note, that this is a naive bound – indeed, we choose the $k/2$ values that are in $I(T)$, and then we generate the tuple T, by choosing values for each coordinate separately. We thus conclude that

$$M_k(X) = \mathbb{E}\left[X^k\right] \leq n^{k/2}(k/2)^k.$$

Using Lemma 3.3.1, we thus get

$$\mathbb{P}\left[|X| \geq \frac{tk}{2}\sqrt{n}\right] = \mathbb{P}\left[|X| \geq t\left(n^{k/2}(k/2)^k\right)^{1/k}\right] \leq \mathbb{P}\left[|X| \geq t\mathbb{E}\left[|X|^k\right]^{1/k}\right] \leq 1/t^k. \qquad \blacksquare$$

Corollary 3.3.3. *Consider k be an even integer and let $X_1, \ldots, X_n$ be n random independent variables such that $\mathbb{P}[X_i = -1] = \mathbb{P}[X_i = +1] = 1/2$. For $X = \sum_{i=1}^n X_i$, and any k, we have $\mathbb{P}\left[|X| \geq k\sqrt{n}\right] \leq 1/2^k$.*

Observe, that the above proof did not require all the variables to be purely independent – it was enough that they are k-wise independent. We readily get the following.

Definition 3.3.4. Given n random variables $X_1, \ldots, X_n$ they are ***k-wise independent***, if for any k of them (i.e., $i_1 < i_2, \ldots, i_k$), and any k values $x_1, \ldots, x_k$, we have

$$\mathbb{P}\left[\bigcap_{\ell=1}^{k} (X_{i_\ell} = v_\ell)\right] = \prod_{\ell=1}^{k} \mathbb{P}\left[X_{i_\ell} = v_\ell\right].$$

Informally, variables are k-wise independent, if any k of them (on their own) looks totally random.

Lemma 3.3.5. *Consider k be an even integer and let $X_1, \ldots, X_n$ be n random independent variables, that are k-wise independent, such that $\mathbb{P}[X_i = -1] = \mathbb{P}[X_i = +1] = 1/2$. Let $X = \sum_{i=1}^n X_i$. Then, we have*

$$\mathbb{P}\left[|X| \geq \frac{tk}{2}\sqrt{n}\right] \leq \frac{1}{t^k}.$$

Chapter 4

Min Cut

To acknowledge the corn - This purely American expression means to admit the losing of an argument, especially in regard to a detail; to retract; to admit defeat. It is over a hundred years old. Andrew Stewart, a member of Congress, is said to have mentioned it in a speech in 1828. He said that haystacks and cornfields were sent by Indiana, Ohio and Kentucky to Philadelphia and New York. Charles A. Wickliffe, a member from Kentucky questioned the statement by commenting that haystacks and cornfields could not walk. Stewart then pointed out that he did not mean literal haystacks and cornfields, but the horses, mules, and hogs for which the hay and corn were raised. Wickliffe then rose to his feet, and said, "Mr. Speaker, I acknowledge the corn".

Funk, Earle, A Hog on Ice and Other Curious Expressions

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
December 10, 2019

4.1. Branching processes – Galton-Watson Process

4.1.1. The problem

In the 19th century, Victorians were worried that aristocratic surnames were disappearing, as family names passed on only through the male children. As such, a family with no male children had its family name disappear. So, imagine the number of male children of a person is an independent random variable $X \in \{0, 1, 2, \ldots\}$. Starting with a single person, its family (as far as male children are concerned) is a random tree with the degree of a node being distributed according to X. We continue recursively in constructing this tree, again, sampling the number of children for each current leaf according to the distribution of X. It is not hard to see that a family disappears if $\mathbb{E}[X] \leq 1$, and it has a constant probability of surviving if $\mathbb{E}[X] > 1$.

Francis Galton asked the question of what is the probability of such a blue-blood family name to survive, and this question was answered by Henry William Watson [WG75]. The Victorians were worried about strange things, see [Gre69] for a provocatively titled article from the period, and [Ste12] for a more recent take on this issue.

Of course, since infant mortality is dramatically down (as is the number of aristocrat males dying to maintain the British empire), the probability of family names to disappear is now much lower than it was in the 19th century. Interestingly, countries with family names that were introduced long time ago have very few surnames (i.e., Korean have 250 surnames, and three surnames form 45% of the population). On the other hand, countries that introduced surnames more recently have dramatically more surnames

(for example, the Dutch have surnames only for the last 200 years, and there are 68,000 different family names).

Here we are going to look on a very specific variant of this problem. Imagine that starting with a single male. A male has exactly two children, and one of them is a male with probability half (i.e., the Y-chromosome is being passed only to its male children). As such, the natural question is what is the probability that h generations down, there is a male decedent that all his ancestors are male (i.e., it caries the original family name, and the original Y-chromosome).

4.1.2. On coloring trees

Let T_h be a complete binary tree of height h. We randomly color its edges by black and white. Namely, for each edge we independently choose its color to be either black or white, with equal probability (say, black indicates the child is male). We are interested in the event that there exists a path from the root of T_h to one of its leafs, that is all black. Let $\mathcal{E}_h$ denote this event, and let $\rho_h = \mathbb{P}[\mathcal{E}_h]$. Observe that $\rho_0 = 1$ and $\rho_1 = 3/4$ (see below).

To bound this probability, consider the root u of T_h and its two children u_l and u_r. The probability that there is a black path from u_l to one of its children is ρ_{h-1}, and as such, the probability that there is a black path from u through u_l to a leaf of the subtree of u_l is $\mathbb{P}[\text{the edge } uu_l \text{ is colored black}] \cdot \rho_{h-1} = \rho_{h-1}/2$. As such, the probability that there is no black path through u_l is $1 - \rho_{h-1}/2$. As such, the probability of not having a black path from u to a leaf (through either children) is $(1 - \rho_{h-1}/2)^2$. In particular, there desired probability, is the complement; that is

$$\rho_h = 1 - \left(1 - \frac{\rho_{h-1}}{2}\right)^2 = \frac{\rho_{h-1}}{2}\left(2 - \frac{\rho_{h-1}}{2}\right) = \rho_{h-1} - \frac{\rho_{h-1}^2}{4} = f(\rho_{h-1}) \qquad \text{for} \qquad f(x) = x - x^2/4.$$

The starting values are $\rho_0 = 1$, and $\rho_1 = 3/4$.

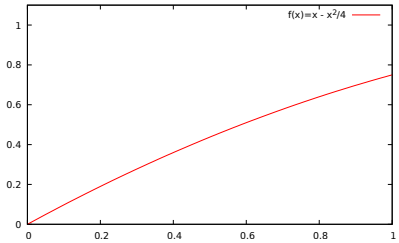


Figure 4.1: A graph of the function $f(x) = x - x^2/4$.

Lemma 4.1.1. *We have that* $\rho_h \geq 1/(h+1)$.

Proof: (Feel free to skip reading.) The proof is by induction. For $h = 1$, we have $\rho_1 = 3/4 \geq 1/(1+1)$.

Observe that $\rho_h = f(\rho_{h-1})$ for $f(x) = x - x^2/4$, and $f'(x) = 1 - x/2$. As such, $f'(x) > 0$ for $x \in [0,1]$ and $f(x)$ is increasing in the range $[0,1]$. As such, by induction, we have that

$$\rho_h = f(\rho_{h-1}) \geq f\left(\frac{1}{(h-1)+1}\right) = \frac{1}{h} - \frac{1}{4h^2}.$$

We need to prove that $\rho_h \geq 1/(h+1)$, which is implied by the above if

$$\frac{1}{h} - \frac{1}{4h^2} \geq \frac{1}{h+1} \quad \Leftrightarrow \quad 4h(h+1) - (h+1) \geq 4h^2 \quad \Leftrightarrow \quad 4h^2 + 4h - h - 1 \geq 4h^2 \quad \Leftrightarrow \quad 3h \geq 1,$$

which trivially holds. ∎

Lemma 4.1.2. *We have that* $\rho_h = O(1/h)$.

Proof: (Feel free to skip reading.) The claim trivially holds for small values of h. For any $j > 0$, let h_j be the minimal index such that $\rho_{h_j} \le 1/2^j$. It is easy to verify that $\rho_{h_j} \ge 1/2^{j+1}$. We claim (mysteriously) that $h_{j+1} - h_j \le \dfrac{\rho_{h_j} - \rho_{h_{j+1}}}{(\rho_{h_{j+1}})^2/4}$ Indeed, ρ_{k+1} is the number resulting from removing $\rho_k^2/4$ from ρ_k. Namely, the sequence $\rho_1, \rho_2, \ldots$ is a monotonically decreasing sequence of numbers in the interval $[0,1]$, where the gaps between consecutive numbers decreases. In particular, to get from ρ_{h_j} to $\rho_{h_{j+1}}$, the gaps used were of size at least $\Delta = \left(\rho_{h_{j+1}}\right)^2$, which means that there are at least $(\rho_{h_j} - \rho_{h_{j+1}})/\Delta - 1$ numbers in the series between these two elements. As such, we have

$$h_{j+1} - h_j \le \frac{\rho_{h_j} - \rho_{h_{j+1}}}{(\rho_{h_{j+1}})^2/4} \le \frac{1/2^j - 1/2^{j+2}}{1/2^{2(j+2)+2}} = 2^{j+6} + 2^{j+4} = O\left(2^j\right).$$

Arguing similarly, we have

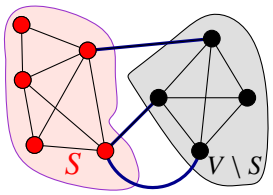
$$h_{j+2} - h_j \ge \frac{\rho_{h_j} - \rho_{h_{j+2}}}{(\rho_{h_j})^2/4} \ge \frac{1/2^{j+1} - 1/2^{j+2}}{1/2^{2j+2}} = 2^{j+1} + 2^j = \Omega\left(2^j\right).$$

We conclude that $h_j = (h_j - h_{j-2}) + (h_{j-2} - h_{j-4}) + \cdots = 2^{j-1} - O(1)$, implying the claim. ■

4.2. Min Cut

4.2.1. Problem Definition

Let $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ be an undirected graph with n vertices and m edges. We are interested in ***cuts*** in G.



Definition 4.2.1. A ***cut*** in G is a partition of the vertices of V into two sets S and $\mathsf{V} \setminus S$, where the edges of the cut are

$$(S, \mathsf{V} \setminus S) = \left\{ uv \;\middle|\; u \in S, v \in \mathsf{V} \setminus S, \text{ and } uv \in E \right\},$$

where $S \neq \emptyset$ and $\mathsf{V} \setminus S \neq \emptyset$. We will refer to the number of edges in the cut $(S, \mathsf{V} \setminus S)$ as the *size of the cut*. For an example of a cut, see figure on the right.

We are interested in the problem of computing the ***minimum cut*** (i.e., ***mincut***), that is, the cut in the graph with minimum cardinality. Specifically, we would like to find the set $S \subseteq \mathsf{V}$ such that $(S, \mathsf{V} \setminus S)$ is as small as possible, and S is neither empty nor $\mathsf{V} \setminus S$ is empty.

4.2.2. Some Definitions

We remind the reader of the following concepts. The ***conditional probability*** of X given Y is $\mathbb{P}\left[X = x \,\middle|\, Y = y\right] = \mathbb{P}[(X = x) \cap (Y = y)]/\mathbb{P}[Y = y]$. An equivalent, useful restatement of this is that

$$\mathbb{P}\left[(X = x) \cap (Y = y)\right] = \mathbb{P}\left[X = x \,\middle|\, Y = y\right] \cdot \mathbb{P}[Y = y]. \tag{4.1}$$

The following is easy to prove by induction using Eq. (4.1).

Lemma 4.2.2. *Let* $\mathcal{E}_1, \ldots, \mathcal{E}_n$ *be* n *events which are not necessarily independent. Then,*

$$\mathbb{P}\left[\cap_{i=1}^n \mathcal{E}_i\right] = \mathbb{P}\left[\mathcal{E}_1\right] * \mathbb{P}\left[\mathcal{E}_2 \,\middle|\, \mathcal{E}_1\right] * \mathbb{P}\left[\mathcal{E}_3 \,\middle|\, \mathcal{E}_1 \cap \mathcal{E}_2\right] * \ldots * \mathbb{P}\left[\mathcal{E}_n \,\middle|\, \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{n-1}\right].$$

4.3. The Algorithm

The basic operation used by the algorithm is ***edge contraction***, depicted in Figure 4.2. We take an edge $e = xy$ in G and merge the two vertices into a single vertex. The new resulting graph is denoted by G/xy. Note, that we remove self loops created by the contraction. However, since the resulting graph is no longer a regular graph, it has parallel edges – namely, it is a multi-graph. We represent a multi-graph, as a regular graph with multiplicities on the edges. See Figure 4.3.

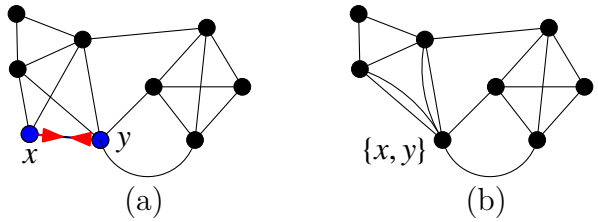


Figure 4.2: (a) A contraction of the edge xy. (b) The resulting graph.

The edge contraction operation can be implemented in $O(n)$ time for a graph with n vertices. This is done by merging the adjacency lists of the two vertices being contracted, and then using hashing to do the fix-ups (i.e., we need to fix the adjacency list of the vertices that are connected to the two vertices).

Note, that the cut is now computed counting multiplicities (i.e., if e is in the cut and it has weight w, then the contribution of e to the cut weight is w).

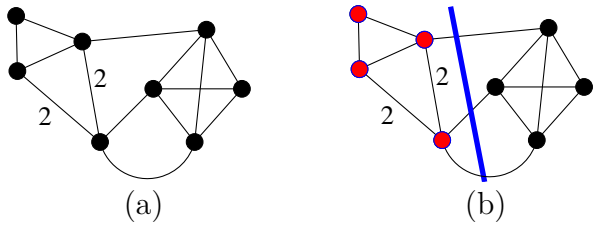


Figure 4.3: (a) A multi-graph. (b) A minimum cut in the resulting multi-graph.

Observation 4.3.1. *A set of vertices in G/xy corresponds to a set of vertices in the graph G. Thus a cut in G/xy always corresponds to a valid cut in G. However, there are cuts in G that do not exist in G/xy. For example, the cut $S = \{x\}$, does not exist in G/xy. As such, the size of the minimum cut in G/xy is at least as large as the minimum cut in G (as long as G/xy has at least one edge). Since any cut in G/xy has a corresponding cut of the same cardinality in G.*

Our algorithm works by repeatedly performing edge contractions. This is beneficial as this shrinks the underlying graph, and we would compute the cut in the resulting (smaller) graph. An "extreme" example of this, is shown in Figure 4.4, where we contract the graph into a single edge, which (in turn) corresponds to a cut in the original graph. (It might help the reader to think about each vertex in the contracted graph, as corresponding to a connected component in the original graph.)

Figure 4.4 also demonstrates the problem with taking this approach. Indeed, the resulting cut is not the minimum cut in the graph.

So, why did the algorithm fail to find the minimum cut in this case?[①] The failure occurs because of the contraction at Figure 4.4 (e), as we had contracted an edge in the minimum cut. In the new graph, depicted in Figure 4.4 (f), there is no longer a cut of size 3, and all cuts are of size 4 or more. Specifically, the algorithm succeeds only if it does not contract an edge in the minimum cut.

Observation 4.3.2. *Let $e_1, \ldots, e_{n-2}$ be a sequence of edges in G, such that none of them is in the minimum cut, and such that $\mathsf{G}' = G/\{e_1, \ldots, e_{n-2}\}$ is a single multi-edge. Then, this multi-edge corresponds to a minimum cut in G.*

Note, that the claim in the above observation is only in one direction. We might be able to still compute a minimum cut, even if we contract an edge in a minimum cut, the reason being that a minimum

[①] Naturally, if the algorithm had succeeded in finding the minimum cut, this would have been **our** success.

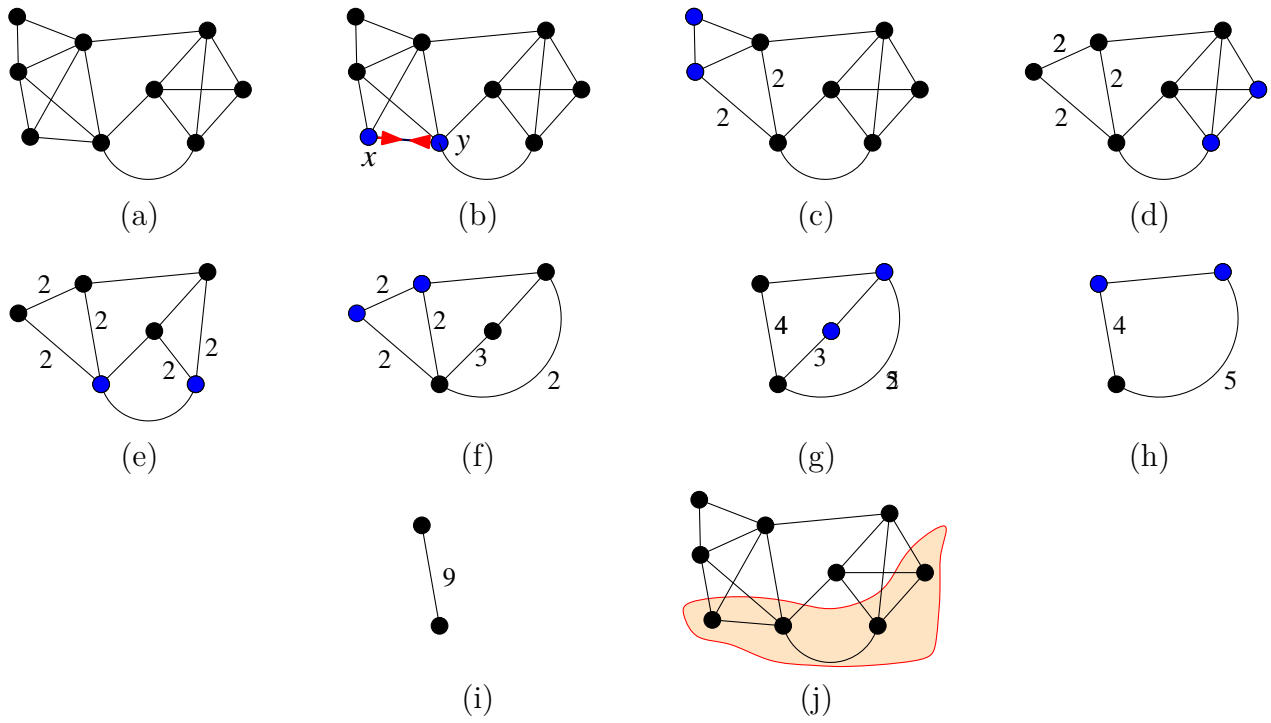


Figure 4.4: (a) Original graph. (b)–(j) a sequence of contractions in the graph, and (h) the cut in the original graph, corresponding to the single edge in (h). Note that the cut of (h) is not a mincut in the original graph.

Algorithm MinCut(G)
$\mathsf{G}_0 \leftarrow G$
$i = 0$
while G_i has more than two vertices **do**
  Pick randomly an edge e_i from the edges of G_i
  $\mathsf{G}_{i+1} \leftarrow G_i / e_i$
  $i \leftarrow i + 1$
Let $(S, \mathsf{V} \setminus S)$ be the cut in the original graph
  corresponding to the single edge in G_i
return $(S, \mathsf{V} \setminus S)$.

Figure 4.5: The minimum cut algorithm.

cut is not unique. In particular, another minimum cut might survived the sequence of contractions that destroyed other minimum cuts.

Using Observation 4.3.2 in an algorithm is problematic, since the argumentation is circular, how can we find a sequence of edges that are not in the cut without knowing what the cut is? The way to slice the Gordian knot here, is to randomly select an edge at each stage, and contract this random edge.

See Figure 4.5 for the resulting algorithm **MinCut**.

4.3.1. Analysis

4.3.1.1. The probability of success

Naturally, if we are extremely lucky, the algorithm would never pick an edge in the mincut, and the algorithm would succeed. The ultimate question here is what is the probability of success. If it is relatively "large" then this algorithm is useful since we can run it several times, and return the best result computed. If on the other hand, this probability is tiny, then we are working in vain since this approach would not work.

Lemma 4.3.3. *If a graph G has a minimum cut of size k and G has n vertices, then $|E(G)| \geq \frac{kn}{2}$.*

Proof: Each vertex degree is at least k, otherwise the vertex itself would form a minimum cut of size smaller than k. As such, there are at least $\sum_{v \in \mathsf{V}} \text{degree(v)}/2 \geq nk/2$ edges in the graph. ■

Lemma 4.3.4. *Fix a specific minimum cut $C = (S, \overline{S})$ in the graph. If we pick in random an edge e from a graph G, uniformly at random, then with probability at most $2/n$ it belongs to the minimum cut C.*

Proof: There are at least $nk/2$ edges in the graph and exactly k edges in the minimum cut. Thus, the probability of picking an edge from the minimum cut is smaller then $k/(nk/2) = 2/n$. ■

The following lemma shows (surprisingly) that **MinCut** succeeds with reasonable probability.

Lemma 4.3.5. **MinCut** *outputs the mincut with probability* $\geq \dfrac{2}{n(n-1)}$.

Proof: Let $\mathcal{E}_i$ be the event that e_i is not in the minimum cut of G_i. By Observation 4.3.2, **MinCut** outputs the minimum cut if the events $\mathcal{E}_0, \ldots, \mathcal{E}_{n-3}$ all happen (namely, all edges picked are outside the minimum cut).

By Lemma 4.3.4, it holds $\mathbb{P}\left[\mathcal{E}_i \,\middle|\, \mathcal{E}_0 \cap \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{i-1}\right] \geq 1 - \dfrac{2}{|\mathsf{V}(G_i)|} = 1 - \dfrac{2}{n-i}$. Implying that

$$\Delta = \mathbb{P}[\mathcal{E}_0 \cap \ldots \cap \mathcal{E}_{n-3}] = \mathbb{P}[\mathcal{E}_0] \cdot \mathbb{P}\left[\mathcal{E}_1 \,\middle|\, \mathcal{E}_0\right] \cdot \mathbb{P}\left[\mathcal{E}_2 \,\middle|\, \mathcal{E}_0 \cap \mathcal{E}_1\right] \cdot \ldots \cdot \mathbb{P}\left[\mathcal{E}_{n-3} \,\middle|\, \mathcal{E}_0 \cap \ldots \cap \mathcal{E}_{n-4}\right].$$

As such, we have

$$\begin{aligned}
\Delta &\geq \prod_{i=0}^{n-3} \left(1 - \frac{2}{n-i}\right) = \prod_{i=0}^{n-3} \frac{n-i-2}{n-i} \\
&= \frac{n-2}{n} * \frac{n-3}{n-1} * \frac{n-4}{n-2} * \frac{n-5}{n-3} * \frac{n-6}{n-4} * \cdots * \frac{3}{5} * \frac{2}{4} * \frac{1}{3} \\
&= \frac{2}{n(n-1)}.
\end{aligned}$$

■

4.3.1.2. Running time analysis.

Observation 4.3.6. **MinCut** *runs in* $O(n^2)$ *time.*

Observation 4.3.7. *The algorithm always outputs a cut, and the cut is not smaller than the minimum cut.*

Definition 4.3.8. (informal) Amplification is the process of running an experiment again and again till the things we want to happen, with good probability, do happen.

Let **MinCutRep** be the algorithm that runs **MinCut** $n(n-1)$ times and return the minimum cut computed in all those independent executions of **MinCut**.

Lemma 4.3.9. *The probability that* **MinCutRep** *fails to return the minimum cut is* < 0.14.

Proof: The probability of failure of **MinCut** to output the mincut in each execution is at most $1-\frac{2}{n(n-1)}$, by Lemma 4.3.5. Now, **MinCutRep** fails, only if all the $n(n-1)$ executions of **MinCut** fail. But these executions are independent, as such, the probability to this happen is at most

$$\left(1-\frac{2}{n(n-1)}\right)^{n(n-1)} \leq \exp\left(-\frac{2}{n(n-1)}\cdot n(n-1)\right) = \exp(-2) < 0.14,$$

since $1-x \leq e^{-x}$ for $0 \leq x \leq 1$. ∎

Theorem 4.3.10. *One can compute the minimum cut in* $O(n^4)$ *time with constant probability to get a correct result. In* $O\left(n^4 \log n\right)$ *time the minimum cut is returned with high probability.*

4.4. A faster algorithm

The algorithm presented in the previous section is extremely simple. Which raises the question of whether we can get a faster algorithm[②]?

So, why **MinCutRep** needs so many executions? Well, the probability of success in the first ν iterations is

$$\begin{aligned}\mathbb{P}[\mathcal{E}_0 \cap \ldots \cap \mathcal{E}_{\nu-1}] &\geq \prod_{i=0}^{\nu-1}\left(1-\frac{2}{n-i}\right) = \prod_{i=0}^{\nu-1}\frac{n-i-2}{n-i} \\ &= \frac{n-2}{n} * \frac{n-3}{n-1} * \frac{n-4}{n-2}\ldots = \frac{(n-\nu)(n-\nu-1)}{n\cdot(n-1)}. \end{aligned} \tag{4.2}$$

Namely, this probability deteriorates very quickly toward the end of the execution, when the graph becomes small enough. (To see this, observe that for $\nu = n/2$, the probability of success is roughly $1/4$, but for $\nu = n-\sqrt{n}$ the probability of success is roughly $1/n$.)

So, the key observation is that as the graph get smaller the probability to make a bad choice increases. So, instead of doing the amplification from the outside of the algorithm, we will run the new algorithm more times when the graph is smaller. Namely, we put the amplification directly into the algorithm.

The basic new operation we use is **Contract**, depicted in Figure 4.6, which also depict the new algorithm **FastCut**.

[②]This would require a more involved algorithm, thats life.

```
Contract ( G, t )
begin
  while |(G)| > t do
      Pick a random edge e in G.
      G ← G/e
  return G
end
```

```
FastCut(G = (V, E))
  G – multi-graph
begin
  n ← |V(G)|
  if n ≤ 6 then
      Compute (via brute force) minimum cut
      of G and return cut.
  t ← ⌈1 + n/√2⌉
  H1 ← Contract(G, t)
  H2 ← Contract(G, t)
  /* Contract is randomized!!! */
  X1 ← FastCut(H1),
  X2 ← FastCut(H2)
  return minimum cut out of X1 and X2.
end
```

Figure 4.6: **Contract**(G,t) shrinks G till it has only t vertices. **FastCut** computes the minimum cut using **Contract**.

Lemma 4.4.1. *The running time of* **FastCut**(G) *is* $O\left(n^2 \log n\right)$*, where* $n = |V(G)|$.

Proof: Well, we perform two calls to **Contract**(G,t) which takes $O(n^2)$ time. And then we perform two recursive calls on the resulting graphs. We have

$$T(n) = O(n^2) + 2T\left(n/\sqrt{2}\right).$$

The solution to this recurrence is $O\left(n^2 \log n\right)$ as one can easily (and should) verify. ■

Exercise 4.4.2. Show that one can modify **FastCut** so that it uses only $O(n^2)$ space.

Lemma 4.4.3. *The probability that* **Contract**$\left(\mathsf{G}, n/\sqrt{2}\right)$ *had not contracted the minimum cut is at least* $1/2$.

Namely, the probability that the minimum cut in the contracted graph is still a minimum cut in the original graph is at least $1/2$.

Proof: Just plug in $\nu = n - t = n - \left\lceil 1 + n/\sqrt{2} \right\rceil$ into Eq. (4.2). We have

$$\mathbb{P}\left[\mathcal{E}_0 \cap \ldots \cap \mathcal{E}_{n-t}\right] \geq \frac{t(t-1)}{n \cdot (n-1)} = \frac{\left\lceil 1 + n/\sqrt{2} \right\rceil \left(\left\lceil 1 + n/\sqrt{2} \right\rceil - 1\right)}{n(n-1)} \geq \frac{1}{2}.$$ ■

The following lemma bounds the probability of success.

Lemma 4.4.4. **FastCut** *finds the minimum cut with probability larger than* $\Omega(1/\log n)$.

Proof: Let T_h be the recursion tree of the algorithm of depth $h = \Theta(\log n)$. Color an edge of recursion tree by black if the contraction succeeded. Clearly, the algorithm succeeds if there is a path from the root to a leaf that is all black. This is exactly the settings of Lemma 4.1.1, and we conclude that the probability of success is at least $1/(h+1) = \Theta(1/\log n)$, as desired. ■

Exercise 4.4.5. Prove, that running **FastCut** repeatedly $c \cdot \log^2 n$ times, guarantee that the algorithm outputs the minimum cut with probability $\geq 1 - 1/n^2$, say, for c a constant large enough.

Theorem 4.4.6. *One can compute the minimum cut in a graph* G *with* n *vertices in* $O(n^2 \log^3 n)$ *time. The algorithm succeeds with probability* $\geq 1 - 1/n^2$.

Proof: We do amplification on **FastCut** by running it $O(\log^2 n)$ times. The running time bound follows from Lemma 4.4.1. The bound on the probability follows from Lemma 4.4.4, and using the amplification analysis as done in Lemma 4.3.9 for **MinCutRep**. ■

4.5. Bibliographical Notes

The **MinCut** algorithm was developed by David Karger during his PhD thesis in Stanford. The fast algorithm is a joint work with Clifford Stein. The basic algorithm of the mincut is described in [MR95, pages 7–9], the faster algorithm is described in [MR95, pages 289–295].

4.5.0.0.1. Galton-Watson process. The idea of using coloring of the edges of a tree to analyze **FastCut** might be new (i.e., Section 4.1.2).

Chapter 5

Hashing

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
December 10, 2019

> "I tried to read this book, Huckleberry Finn, to my grandchildren, but I couldn't get past page six because the book is fraught with the 'n-word.' And although they are the deepest-thinking, combat-ready eight- and ten-year-olds I know, I knew my babies weren't ready to comprehend Huckleberry Finn on its own merits. That's why I took the liberty to rewrite Mark Twain's masterpiece. Where the repugnant 'n-word' occurs, I replaced it with 'warrior' and the word 'slave' with 'dark-skinned volunteer."'
>
> Paul Beatty, The Sellout

5.1. Introduction

We are interested here in dictionary data structure. The settings for such a data-structure:

(A) $\mathcal{U}$: universe of keys with total order: numbers, strings, etc.
(B) Data structure to store a subset $S \subseteq \mathcal{U}$
(C) **Operations:**
  (A) **search/lookup**: given $x \in \mathcal{U}$ is $x \in S$?
  (B) **insert**: given $x \notin S$ add x to S.
  (C) **delete**: given $x \in S$ delete x from S
(D) ***Static*** structure: S given in advance or changes very infrequently, main operations are lookups.
(E) ***Dynamic*** structure: S changes rapidly so inserts and deletes as important as lookups.

Common constructions for such data-structures, include using a static sorted array, where the lookup is a binary search. Alternatively, one might use a *balanced* search tree (i.e., red-black tree). The time to perform an operation like lookup, insert, delete take $O(\log |S|)$ time (comparisons).

Naturally, the above are potently an "overkill", in the sense that sorting is unnecessary. In particular, the universe $\mathcal{U}$ may not be (naturally) totally ordered. The keys correspond to large objects (images, graphs etc) for which comparisons are expensive. Finally, we would like to improve "average" performance of lookups to $O(1)$ time, even at cost of extra space or errors with small probability: many applications for fast lookups in networking, security, etc.

Hashing and Hash Tables. The hash-table data structure has an associated (hash) table/array T of size m (the table ***size***). A hash function $h : \mathcal{U} \to \{0, \ldots, m-1\}$. An item $x \in \mathcal{U}$ hashes to slot $h(x)$ in T.

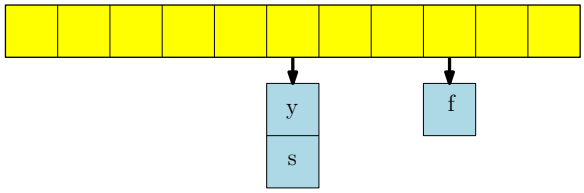


Figure 5.1: Open hashing.

Given a set $S \subseteq \mathcal{U}$, in a perfect ideal situation, each element $x \in S$ hashes to a distinct slot in T, and we store x in the slot $h(x)$. The **Lookup** for an item $y \in \mathcal{U}$, is to check if $T[h(y)] = y$. This takes constant time.

Unfortunately, *collisions* are unavoidable, and several different techniques to handle them. Formally, two items $x \neq y$ ***collide*** if $h(x) = h(y)$.

A standard technique to handle collisions is to use ***chaining*** (aka *open hashing*). Here, we handle collisions as follows:

(A) For each slot i store all items hashed to slot i in a linked list. $T[i]$ points to the linked list.

(B) **Lookup**: to find if $y \in \mathcal{U}$ is in T, check the linked list at $T[h(y)]$. Time proportion to size of linked list.

Other techniques for handling collisions include associating a list of locations where an element can be (in certain order), and check these locations in this order. Another useful technique is ***cuckoo hashing*** which we will discuss later on: Every value has two possible locations. When inserting, insert in one of the locations, otherwise, kick out the stored value to its other location. Repeat till stable. if no stability then rebuild table.

The relevant questions when designing a hashing scheme, include: (I) Does hashing give $O(1)$ time per operation for dictionaries? (II) Complexity of evaluating h on a given element? (III) Relative sizes of the universe $\mathcal{U}$ and the set to be stored S. (IV) Size of table relative to size of S. (V) Worst-case vs average-case vs randomized (expected) time? (VI) How do we choose h?

The ***load factor*** of the array T is the ratio n/t where $n = |S|$ is the number of elements being stored and $m = |T|$ is the size of the array being used. Typically n/t is a small constant smaller than 1.

In the following, we assume that $\mathcal{U}$ (the universe the keys are taken from) is large – specifically, $N = |\mathcal{U}| \gg m^2$, where m is the size of the table. Consider a hash function $h : \mathcal{U} \to \{0, \ldots, m-1\}$. If hash N items to the m slots, then by the pigeon hole principle, there is some $i \in \{0, \ldots, m-1\}$ such that $N/m \geq m$ elements of $\mathcal{U}$ get hashed to i. In particular, this implies that there is set $S \subseteq \mathcal{U}$, where $|S| = m$ such that all of S hashes to same slot. Oops.

Namely, for every hash function there is a *bad set* with many collisions.

Observation 5.1.1. *Let $\mathcal{H}$ be the set of all functions from $\mathcal{U} = \{1, \ldots, U\}$ to $\{1, \ldots, m\}$. The number of functions in $\mathcal{H}$ is m^U. As such, specifying a function in $\mathcal{H}$ would require $\log_2 |\mathcal{H}| = O(U \log m)$.*

As such, picking a truely random hash function requires many random bits, and furthermore, it is not even clear how to evaluate it efficiently (which is the whole point of hashing).

Picking a hash function. Picking a good hash function in practice is a dark art involving many non-trivial considerations and ideas. For parameters $N = |\mathcal{U}|$, $m = |T|$, and $n = |S|$, we require the following:

(A) $\mathcal{H}$ is a ***family*** of hash functions: each function $h \in \mathcal{H}$ should be efficient to evaluate (that is, to compute $h(x)$).

(B) h is chosen *randomly* from $\mathcal{H}$ (typically uniformly at random). Implicitly assumes that $\mathcal{H}$ allows an efficient sampling.

(C) Require that for any *fixed* set $S \subseteq \mathcal{U}$, of size m, the expected number of collisions for a function chosen from $\mathcal{H}$ should be "small". Here the expectation is over the randomness in choice of h.

5.2. Universal Hashing

We would like the hash function to have the following property – For any element $x \in \mathcal{U}$, and a random $h \in \mathcal{H}$, then $h(x)$ should have a uniform distribution. That is $\Pr[h(x) = i] = 1/m$, for every $0 \le i < m$. A somewhat stronger property is that for any two distinct elements $x, y \in \mathcal{U}$, for a random $h \in \mathcal{H}$, the probability of a collision between x and y should be at most $1/m$. $\mathbb{P}[h(x) = h(y)] = 1/m$.

Definition 5.2.1. A family $\mathcal{H}$ of hash functions is ***2-universal*** if for all distinct $x, y \in \mathcal{U}$, we have $\mathbb{P}[h(x) = h(y)] \le 1/m$.

Applying a 2-universal family hash function to a set of distinct numbers, results in a 2-wise independent sequence of numbers.

Lemma 5.2.2. *Let S be a set of n elements stored using open hashing in a hash table of size m, using open hashing, where the hash function is picked from a 2-universal family. Then, the expected lookup time, for any element $x \in \mathcal{U}$ is $O(n/m)$.*

Proof: The number of elements colliding with x is $\ell(x) = \sum_{y \in S} D_y$, where $D_y = 1 \iff x$ and y collide under the hash function h. As such, we have

$$\mathbb{E}[\ell(x)] = \sum_{y \in S} \mathbb{E}\big[D_y\big] = \sum_{y \in S} \mathbb{P}[h(x) = h(y)] = \sum_{y \in S} \frac{1}{m} = |S|/m = n/m. \qquad \blacksquare$$

Remark 5.2.3. The above analysis holds even if we perform a sequence of $O(n)$ insertions/deletions operations. Indeed, just repeat the analysis with the set of elements being all elements encountered during these operations.

The worst-case bound is of course much worse – it is not hard to show that in the worst case, the load of a single hash table entry might be $\Omega(\log n / \log \log n)$ (as we seen in the occupancy problem).

Rehashing, amortization, etc. The above assumed that the set S is fixed. If items are inserted and deleted, then the hash table might become much worse. In particular, $|S|$ grows to more than cm, for some constant c, then hash table performance start degrading. Furthermore, if many insertions and deletions happen then the initial random hash function is no longer random enough, and the above analysis no longer holds.

A standard solution is to rebuild the hash table periodically. We choose a new table size based on current number of elements in table, and a new random hash function, and rehash the elements. And then discard the old table and hash function. In particular, if $|S|$ grows to more than twice current table size, then rebuild new hash table (choose a new random hash function) with double the current number of elements. One can do a similar shrinking operation if the set size falls below quarter the current hash table size.

If the working $|S|$ stays roughly the same but more than $c|S|$ operations on table for some chosen constant c (say 10), rebuild.

The ***amortize*** cost of rebuilding to previously performed operations. Rebuilding ensures $O(1)$ expected analysis holds even when S changes. Hence $O(1)$ expected look up/insert/delete time *dynamic* data dictionary data structure!

5.2.1. How to build a 2-universal family

5.2.1.1. A quick reminder on working modulo prime

Definition 5.2.4. For a number p, let $\mathbb{Z}_n = \{0, \ldots, n-1\}$.

For two integer numbers x and y, the ***quotient*** of x/y is $x \operatorname{div} y = \lfloor x/y \rfloor$. The ***remainder*** of x/y is $x \bmod y = x - y\lfloor x/y \rfloor$. If the $x \bmod y = 0$, than y ***divides*** x, denoted by $y \mid x$. We use $\alpha \equiv \beta \pmod{p}$ or $\alpha \equiv_p \beta$ to denote that α and β are ***congruent modulo*** p; that is $\alpha \bmod p = \beta \bmod p$ – equivalently, $p \mid (\alpha - \beta)$.

Remark 5.2.5. A quick review of what we already know. Let p be a prime number.

(A) Lemma 3.2.3: For any $\alpha, \beta \in \{1, \ldots, p-1\}$, we have that $\alpha\beta \not\equiv 0 \pmod{p}$.
(B) Lemma 3.2.3: For any $\alpha, \beta, i \in \{1, \ldots, p-1\}$, such that $\alpha \neq \beta$, we have that $\alpha i \not\equiv \beta i \pmod{p}$.
(C) Lemma 3.2.3: For any $x \in \{1, \ldots, p-1\}$ there exists a unique y such that $xy \equiv 1 \pmod{p}$. The number y is the ***inverse*** of x, and is denoted by x^{-1} or $1/x$.
(D) Lemma 3.2.4: For any numbers $x, y \in \mathbb{Z}_p$. If $x \neq y$ then, for any $a, b \in \mathbb{Z}_p$, such that $a \neq 0$, we have $ax + b \not\equiv ay + b \pmod{p}$.
(E) Lemma 3.2.5: For any numbers $x, y \in \mathbb{Z}_p$. If $x \neq y$ then, for each pair of numbers $r, s \in \mathbb{Z}_p = \{0, 1, \ldots, p-1\}$, such that $r \neq s$, there is exactly one unique choice of numbers $a, b \in \mathbb{Z}_p$ such that $ax + b \pmod{p} = r$ and $ay + b \pmod{p} = s$.

5.2.1.2. Constructing a family of 2-universal hash functions

For parameters $N = |\mathcal{U}|$, $m = |T|$, $n = |S|$. Choose a ***prime*** number $p \geq N$. Let

$$\mathcal{H} = \left\{ h_{a,b} \;\middle|\; a, b \in \mathbb{Z}_p \text{ and } a \neq 0 \right\},$$

where $h_{a,b}(x) = ((ax + b) \pmod{p}) \pmod{m}$. Note that $|\mathcal{H}| = p(p-1)$.

5.2.1.3. Analysis

Once we fix a and b, and we are given a value x, we compute the hash value of x in two stages:
(A) ***Compute***: $r \leftarrow (ax + b) \pmod{p}$.
(B) ***Fold***: $r' \leftarrow r \pmod{m}$

Lemma 5.2.6. *Assume that p is a prime, and $1 < m < p$. The number of pairs $(r, s) \in \mathbb{Z}_p \times \mathbb{Z}_p$, such that $r \neq s$, that are folded to the same number is $\leq p(p-1)/m$. Formally, the set of bad pairs*

$$B = \left\{ (r, s) \in \mathbb{Z}_p \times \mathbb{Z}_p \;\middle|\; r \equiv_m s \right\}$$

is of size at most $p(p-1)/m$.

Proof: Consider a pair $(x, y) \in \{0, 1, \ldots, p-1\}^2$, such that $x \neq y$. For a fixed x, there are at most $\lceil p/m \rceil$ values of y that fold into x. Indeed, $x \equiv_m y$ if and only if

$$y \in L(x) = \{x + im \mid i \text{ is an integer}\} \cap \mathbb{Z}_p.$$

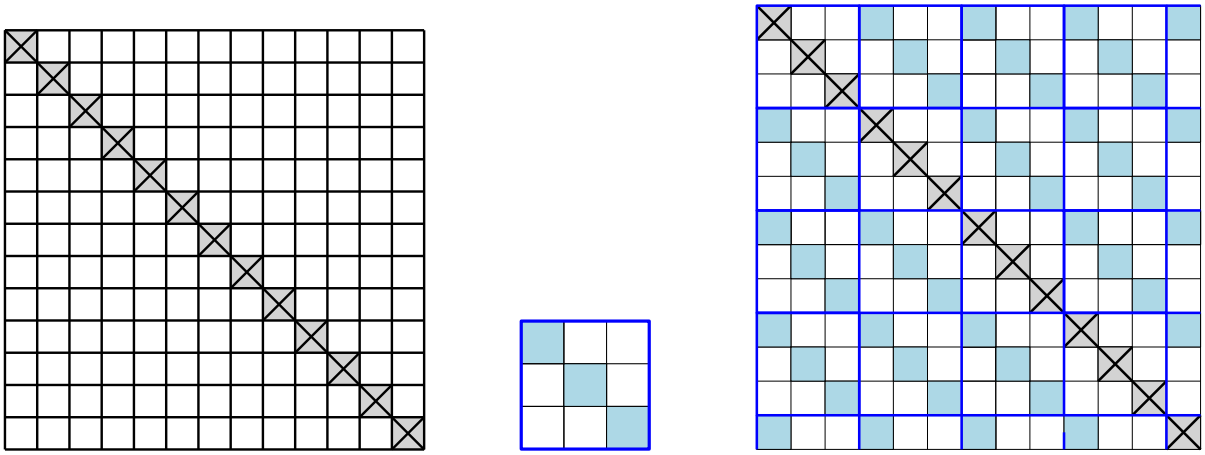


Figure 5.2: Explanation of the hashing scheme via figures.

The size of $L(x)$ is maximized when $x = 0$, The number of such elements is at most $\lceil p/m \rceil$ (note, that since p is a prime, p/m is fractional). One of the numbers in $O(x)$ is x itself. As such, we have that

$$|B| \leq p\big(|L(x)| - 1\big) \leq p\big(\lceil p/m \rceil - 1\big) \leq p\big(p-1\big)/m,$$

since $\lceil p/m \rceil - 1 \leq (p-1)/m \iff m\lceil p/m \rceil - m \leq p - 1 \iff m\lfloor p/m \rfloor \leq p - 1 \iff m\lfloor p/m \rfloor < p$, which is true since p is a prime, and $1 < m < p$. ■

Claim 5.2.7. *For two distinct numbers $x, y \in \mathcal{U}$, a pair a, b is* **bad** *if $h_{a,b}(x) = h_{a,b}(y)$. The number of bad pairs is $\leq p(p-1)/m$.*

Proof: Let $a, b \in \mathbb{Z}_p$ such that $a \neq 0$ and $h_{a,b}(x) = h_{a,b}(y)$. Let

$$r = (ax + b) \bmod p \qquad \text{and} \qquad s = (ay + b) \bmod p.$$

By Lemma 3.2.4, we have that $r \neq s$. As such, a collision happens if $r \equiv s \pmod{m}$. By Lemma 5.2.6, the number of such pairs (r, s) is at most $p(p-1)/m$. By Lemma 3.2.5, for each such pair (r, s), there is a unique choice of a, b that maps x and y to r and s, respectively. As such, there are at most $p(p-1)/m$ bad pairs. ■

Theorem 5.2.8. *The hash family $\mathcal{H}$ is a 2-universal hash family.*

Proof: Fix two distinct numbers $x, y \in \mathcal{U}$. We are interested in the probability they collide if h is picked randomly from $\mathcal{H}$. By Claim 5.2.7 there are $M \leq p(p-1)/m$ bad pairs that causes such a collision, and since $\mathcal{H}$ contains $N = p(p-1)$ functions, it follows the probability for collision is $M/N \leq 1/m$, which implies that $\mathcal{H}$ is 2-universal. ■

5.2.1.4. Explanation via pictures

Consider a pair $(x, y) \in \mathbb{Z}_p^2$, such that $x \neq y$. This pair (x, y) corresponds to a cell in the natural "grid" $\mathbb{Z}_p^2$ that is off the main diagonal. See Figure 5.2

The mapping $f_{a,b}(x) = (ax + b) \bmod p$, takes the pair (x, y), and maps it randomly and uniformly, to some other pair $x' = f_{a,b}(x)$ and $y' = f_{a,b}(y)$ (where x', y' are again off the main diagonal).

Now consider the smaller grid $\mathbb{Z}_m \times \mathbb{Z}_m$. The main diagonal of this subgrid is bad – it corresponds to a collision. One can think about the last step, of computing $h_{a,b}(x) = f_{a,b}(x) \bmod m$, as tiling the larger grid, by the smaller grid. in the natural way. Any diagonal that is in distance mi from the main diagonal get marked as bad. At most $1/m$ fraction of the off diagonal cells get marked as bad. See Figure 5.2.

As such, the random mapping of (x, y) to (x', y') causes a collision only if we map the pair to a badly marked pair, and the probability for that $\leq 1/m$.

5.3. Perfect hashing

An interesting special case of hashing is the static case – given a set S of elements, we want to hash S so that we can answer membership queries efficiently (i.e., dictionary data-structures with no insertions). it is easy to come up with a hashing scheme that is optimal as far as space.

5.3.1. Some easy calculations

The first observation is that if the hash table is quadraticly large, then there is a good (constant) probability to have no collisions (this is also the threshold for the birthday paradox).

Lemma 5.3.1. *Let $S \subseteq \mathcal{U}$ be a set of n elements, and let $\mathcal{H}$ be a 2-universal family of hash functions, into a table of size $m \geq n^2$. Then with probability $\leq 1/2$, there is a pair of elements of S that collide under a random hash function $h \in \mathcal{H}$.*

Proof: For a pair $x, y \in S$, the probability they collide is at most $\leq 1/m$, by definition. As such, by the union bound, the probability of any collusion is $\binom{n}{2}/m = n(n-1)/2m \leq 1/2$. ■

We now need a second moment bound on the sizes of the buckets.

Lemma 5.3.2. *Let $S \subseteq \mathcal{U}$ be a set of n elements, and let $\mathcal{H}$ be a 2-universal family of hash functions, into a table of size $m \geq cn$, where c is an arbitrary constant. Let $h \in \mathcal{H}$ be a random hash function, and let X_i be the number of elements of S mapped to the ith bucket by h, for $i = 0, \ldots, m-1$. Then, we have $\mathbb{E}\left[\sum_{j=0}^{m-1} X_j^2\right] \leq (1 + 1/c)n$.*

Proof: Let $s_1, \ldots, s_n$ be the n items in S, and let $Z_{i,j} = 1$ if $h(s_i) = h(s_j)$, for $i < j$. Observe that $\mathbb{E}\left[Z_{i,j}\right] = \mathbb{P}\left[h(s_i) = h(s_j)\right] \leq 1/m$ (this is the only place we use the property that $\mathcal{H}$ is 2-universal). In particular, let $\mathcal{Z}(\alpha)$ be all the variables $Z_{i,j}$, for $i < j$, such that $Z_{i,j} = 1$ and $h(s_i) = h(s_j) = \alpha$.

If for some α we have that $X_\alpha = k$, then there are k indices $\ell_1 < \ell_2 < \ldots < \ell_k$, such that $h(s_{\ell_1}) = \cdots = h(s_{\ell_k}) = i$. As such, $z(\alpha) = |\mathcal{Z}(\alpha)| = \binom{k}{2}$. In particular, we have

$$X_\alpha^2 = k^2 = 2\binom{k}{2} + k = 2z(\alpha) + X_\alpha$$

This implies that

$$\sum_{\alpha=0}^{m-1} X_\alpha^2 = \sum_{\alpha=0}^{m-1} \left(2z(\alpha) + X_\alpha\right) = 2\sum_{\alpha=0}^{m-1} z(\alpha) + \sum_{\alpha=0}^{m-1} X_\alpha = n + 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} Z_{ij}$$

Now, by linearity of expectations, we have

$$\mathbb{E}\Big[\sum_{\alpha=0}^{m-1} X_\alpha^2\Big] = \mathbb{E}\Big[n + 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} Z_{ij}\Big] = n + 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\mathbb{E}\big[Z_{ij}\big] \le n + 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{1}{m}$$
$$= n + \frac{2}{m}\binom{n}{2} = n + \frac{2n(n-1)}{2m} \le n\left(1 + \frac{n-1}{m}\right) \le n\left(1 + \frac{1}{c}\right)$$

since $m \ge cn$. ■

5.3.2. Construction of perfect hashing

Given a set S of n elements, we build a open hash table T of size, say, $2n$. We use a random hash function h that is 2-universal for this hash table, see Theorem 5.2.8. Next, we map the elements of S into the hash table. Let S_j be the list of all the elements of S mapped to the jth bucket, and let $X_j = \big|L_j\big|$, for $j = 0, \ldots, n-1$.

We compute $Y = \sum_{i=1} X_j^2$. If $Y > 6n$, then we reject h, and resample a hash function h. We repeat this process till success.

In the second stage, we build secondary hash tables for each bucket. Specifically, for $j = 0, \ldots, 2n-1$, if the jth bucket contains $X_j > 0$ elements, then we construct a secondary hash table H_j to store the elements of S_j, and this secondary hash table has size X_j^2, and again we use a random 2-universal hash function h_j for the hashing of S_j into H_j. If any pair of elements of S_j collide under h_j, then we resample the hash function h_j, and try again till success.

5.3.2.1. Analysis

Theorem 5.3.3. *Given a (static) set $S \subseteq \mathcal{U}$ of n elements, the above scheme, constructs, in expected linear time, a two level hash-table that can perform search queries in $O(1)$ time. The resulting data-structure uses $O(n)$ space.*

Proof: Given an element $x \in \mathcal{U}$, we first compute $j = h(x)$, and then $k = h_j(x)$, and we can check whether the element stored in the secondary hash table H_j at the entry k is indeed x. As such, the search time is $O(1)$.

The more interesting issue is the construction time. Let X_j be the number of elements mapped to the jth bucket, and let $Y = \sum_{i=1}^{n} X_i^2$. Observe, that $\mathbb{E}[Y] \le (1 + 1/2)n = (3/2)n$, by Lemma 5.3.2 (here, $m = 2n$ and as such $c = 2$). As such, by Markov's inequality, $\mathbb{P}[X > 6n] = \frac{(3/2)n}{6n} \le 1/4$. In particular, picking a good top level hash function requires in expectation at most $1/(3/4) = 4/3 \le 2$ iterations. Thus the first stage takes $O(n)$ time, in expectation.

For the jth bucket, with X_j entries, by Lemma 5.3.1, the construction succeeds with probability $\ge 1/2$. As before, the expected number of iterations till success is at most 2. As such, the expected construction time of the secondary hash table for the jth bucket is $O(X_j^2)$.

We conclude that the overall expected construction time is $O(n + \sum_j X_j^2) = O(n)$.

As for the space used, observe that it is $O(n + \sum_j X_j^2) = O(n)$. ■

5.4. Bloom filters

Consider an application where we have a set $S \subseteq \mathcal{U}$ of n elements, and we want to be able to decide for a query $x \in \mathcal{U}$, whether or not $x \in S$. Naturally, we can use hashing. However, here we are interested in

more efficient data-structure as far as space. We allow the data-structure to make a mistake (i.e., say that an element is in, when it is not in).

First try. So, let start silly. Let $B[0 \ldots, m]$ be an array of bits, and pick a random hash function $h : \mathcal{U} \to \mathbb{Z}_m$. Initialize B to 0. Next, for every element $s \in S$, set $B[h(s)]$ to 1. Now, given a query, return $B[h(x)]$ as an answer whether or not $x \in S$. Note, that B is an array of bits, and as such it can be bit-packed and stored efficiently.

For the sake of simplicity of exposition, assume that the hash functions picked is truly random. As such, we have that the probability for a false positive (i.e., a mistake) for a fixed $x \in \mathcal{U}$ is n/m. Since we want the size of the table m to be close to n, this is not satisfying.

Using k hash functions. Instead of using a single hash function, let us use k independent hash functions $h_1, \ldots h_k$. For an element $s \in S$, we set $B[h_i(s)]$ to 1, for $i = 1, \ldots, k$. Given an query $x \in \mathcal{U}$, if $B[h_i(x)]$ is zero, for any $i = 1, \ldots, k$, then $x \notin S$. Otherwise, if all these k bits are on, the data-structure returns that x is in S.

Clearly, if the data-structure returns that x is not in S, then it is correct. The data-structure might make a mistake (i.e., a false positive), if it returns that x is in S (when is not in S).

We interpret the storing of the elements of S in B, as an experiment of throwing kn balls into m bins. The probability of a bin to be empty is

$$p = p(m, n) = (1 - 1/m)^{kn} \approx \exp(-k(n/m)).$$

Since the number of empty bins is a martingale, we know the number of empty bins is strongly concentrated around the expectation pm, and we can treat p as the true probability of a bin to be empty.

The probability of a mistake is

$$f(k, m, n) = (1 - p)^k.$$

In particular, for $k = (m/n) \ln n$, we have that $p = p(m, n) \approx 1/2$, and $f(k, m, n) \approx 1/2^{(m/n) \ln 2} \approx 0.618^{m/n}$.

Example 5.4.1. Of course, the above is fictional, as k has to be an integer. But motivated by these calculations, let $m = 3n$, and $k = 4$. We get that $p(m, n) = \exp(-4/3) \approx 0.26359$, and $f(4, 3n, n) \approx (1 - 0.265)^4 \approx 0.294078$. This is better than the naive $k = 1$ scheme, where the probability of false positive is $1/3$.

Note, that this scheme gets exponentially better over the naive scheme as m/n grows.

Example 5.4.2. Consider the setting $m = 8n$ – this is when we allocate a byte for each element stored (the element of course might be significantly bigger). The above implies we should take $k = \lceil (m/n) \ln 2 \rceil = 6$. We then get $p(8n, n) = \exp(-6/8) \approx 0.5352$, and $f(6, 8n, n) \approx 0.0215$. Here, the naive scheme with $k = 1$, would give probability of false positive of $1/8 = 0.125$. So this is a significant improvement.

Remark 5.4.3. It is important to remember that Bloom filters are competing with direct hashing of the whole elements. Even if one allocates 8 bits per item, as in the example above, the space it uses is significantly smaller than regular hashing. A situation when such a Bloom filter makes sense is for a cache – we might want to decide if an element is in a slow external cache (say SSD drive). Retrieving item from the cache is slow, but not so slow we are not willing to have a small overhead because of false positives.

5.5. Bibliographical notes

Practical Issues Hashing used typically for integers, vectors, strings etc.

- Universal hashing is defined for integers. To implement it for other objects, one needs to map objects in some fashion to integers.
- Practical methods for various important cases such as vectors, strings are studied extensively. See http://en.wikipedia.org/wiki/Universal_hashing for some pointers.
- Recent important paper bridging theory and practice of hashing. "The power of simple tabulation hashing" by Mikkel Thorup and Mihai Patrascu, 2011. See http://en.wikipedia.org/wiki/Tabulation_hashing

Chapter 6

Occupancy and Coupon Collector Problems

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
December 10, 2019

I built on the sand
And it tumbled down,
I built on a rock
And it tumbled down.
Now when I build, I shall begin
With the smoke from the chimney

Leopold Staff, Foundations

6.1. Preliminaries

Definition 6.1.1 (Variance and Standard Deviation). For a random variable X, let

$$\mathbb{V}[X] = \mathbb{E}\big[(X - \mu_X)^2\big] = \mathbb{E}\big[X^2\big] - \mu_X^2$$

denote the ***variance*** of X, where $\mu_X = \mathbb{E}[X]$. Intuitively, this tells us how concentrated is the distribution of X. The ***standard deviation*** of X, denoted by σ_X is the quantity $\sqrt{\mathbb{V}[X]}$.

Observation 6.1.2. *(i) For any constant $c \geq 0$, we have $\mathbb{V}\big[cX\big] = c^2\,\mathbb{V}\big[X\big]$.*
(ii) For X and Y independent variables, we have $\mathbb{V}\big[X+Y\big] = \mathbb{V}\big[X\big] + \mathbb{V}\big[Y\big]$.

Definition 6.1.3 (Bernoulli distribution). Assume, that one flips a coin and get 1 (heads) with probability p, and 0 (i.e., tail) with probability $q = 1 - p$. Let X be this random variable. The variable X is has ***Bernoulli distribution*** with parameter p.

We have that $\mathbb{E}[X] = 1 \cdot p + 0 \cdot (1-p) = p$, and

$$\mathbb{V}\big[X\big] = \mathbb{E}\big[X^2\big] - \mu_X^2 = \mathbb{E}\big[X^2\big] - p^2 = p - p^2 = p(1-p) = pq.$$

Definition 6.1.4 (Binomial distribution). Assume that we repeat a Bernoulli experiment n times (independently!). Let $X_1, \ldots, X_n$ be the resulting random variables, and let $X = X_1 + \cdots + X_n$. The variable X has the ***binomial distribution*** with parameters n and p. We denote this fact by $X \sim \mathrm{Bin}(n,p)$. We have

$$b(k;n,p) = \mathbb{P}\big[X = k\big] = \binom{n}{k} p^k q^{n-k}.$$

Also, $\mathbb{E}[X] = np$, and $\mathbb{V}[X] = \mathbb{V}\big[\sum_{i=1}^n X_i\big] = \sum_{i=1}^n \mathbb{V}[X_i] = npq$.

Observation 6.1.5. *Let $C_1, \ldots, C_n$ be random events (not necessarily independent). Than*

$$\mathbb{P}\left[\cup_{i=1}^{n} C_i\right] \leq \sum_{i=1}^{n} \mathbb{P}[C_i].$$

(This is usually referred to as the **union bound**.*) If $C_1, \ldots, C_n$ are* disjoint *events then*

$$\mathbb{P}\left[\cup_{i=1}^{n} C_i\right] = \sum_{i=1}^{n} \mathbb{P}[C_i].$$

6.1.1. Geometric distribution

Definition 6.1.6. Consider a sequence $X_1, X_2, \ldots$ of independent Bernoulli trials with probability p for success. Let X be the number of trials one has to perform till encountering the first success. The distribution of X is ***geometric distribution*** with parameter p. We denote this by $X \sim \mathrm{Geom}(p)$.

Lemma 6.1.7. *For a variable $X \sim \mathrm{Geom}(p)$, we have, for all i, that $\mathbb{P}[X = i] = (1-p)^{i-1}p$. Furthermore, $\mathbb{E}[X] = 1/p$ and $\mathbb{V}[X] = (1-p)/p^2$.*

Proof: The proof of the expectation and variance is included for the sake of completeness, and the reader is of course encouraged to skip (reading) this proof. So, let $f(x) = \sum_{i=0}^{\infty} x^i = \frac{1}{1-x}$, and observe that $f'(x) = \sum_{i=1}^{\infty} i x^{i-1} = (1-x)^{-2}$. As such, we have

$$\mathbb{E}[X] = \sum_{i=1}^{\infty} i\,(1-p)^{i-1}p = pf'(1-p) = \frac{p}{(1-(1-p))^2} = \frac{1}{p},$$

$$\text{and } \mathbb{V}[X] = \mathbb{E}\left[X^2\right] - \frac{1}{p^2} = \sum_{i=1}^{\infty} i^2\,(1-p)^{i-1}p - \frac{1}{p^2}. = p + p(1-p)\sum_{i=2}^{\infty} i^2\,(1-p)^{i-2} - \frac{1}{p^2}.$$

Observe that

$$f''(x) = \sum_{i=2}^{\infty} i(i-1)x^{i-2} = \left((1-x)^{-1}\right)'' = \frac{2}{(1-x)^3}.$$

As such, we have that

$$\begin{aligned}\Delta(x) &= \sum_{i=2}^{\infty} i^2 x^{i-2} = \sum_{i=2}^{\infty} i(i-1)x^{i-2} + \sum_{i=2}^{\infty} i x^{i-2} = f''(x) + \frac{1}{x}\sum_{i=2}^{\infty} i x^{i-1} = f''(x) + \frac{1}{x}(f'(x)-1) \\ &= \frac{2}{(1-x)^3} + \frac{1}{x}\left(\frac{1}{(1-x)^2} - 1\right) = \frac{2}{(1-x)^3} + \frac{1}{x}\left(\frac{1-(1-x)^2}{(1-x)^2}\right) = \frac{2}{(1-x)^3} + \frac{1}{x}\cdot\frac{x(2-x)}{(1-x)^2} \\ &= \frac{2}{(1-x)^3} + \frac{2-x}{(1-x)^2}.\end{aligned}$$

As such, we have that

$$\begin{aligned}\mathbb{V}[X] &= p + p(1-p)\Delta(1-p) - \frac{1}{p^2} = p + p(1-p)\left(\frac{2}{p^3} + \frac{1+p}{p^2}\right) - \frac{1}{p^2} = p + \frac{2(1-p)}{p^2} + \frac{1-p^2}{p} - \frac{1}{p^2} \\ &= \frac{p^3 + 2(1-p) + p - p^3 - 1}{p^2} = \frac{1-p}{p^2}.\end{aligned}$$

■

6.1.2. Some needed math

Lemma 6.1.8. *For any positive integer n, we have:*

(i) $(1+1/n)^n \le e$.

(ii) $(1-1/n)^{n-1} \ge e^{-1}$.

(iii) $n! \ge (n/e)^n$.

(iv) For any $k \le n$, we have: $\left(\frac{n}{k}\right)^k \le \binom{n}{k} \le \left(\frac{ne}{k}\right)^k$.

Proof: (i) Indeed, $1+1/n \le \exp(1/n)$, since $1+x \le e^x$, for $x \ge 0$. As such $(1+1/n)^n \le \exp(n(1/n)) = e$.

(ii) Rewriting the inequality, we need to prove $\left(\frac{n-1}{n}\right)^{n-1} \ge \frac{1}{e}$. This is equivalence to proving $e \ge \left(\frac{n}{n-1}\right)^{n-1} = \left(1+\frac{1}{n-1}\right)^{n-1}$, which is our friend from (i).

(iii) Indeed,

$$\frac{n^n}{n!} \le \sum_{i=0}^{\infty} \frac{n^i}{i!} = e^n,$$

by the Taylor expansion of $e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!}$. This implies that $(n/e)^n \le n!$, as required.

(iv) For any $k \le n$, we have $\frac{n}{k} \le \frac{n-1}{k-1}$ since $kn - n = n(k-1) \le k(n-1) = kn - k$. As such, $\frac{n}{k} \le \frac{n-i}{k-i}$, for $1 \le i \le k-1$. As such,

$$\left(\frac{n}{k}\right)^k \le \frac{n}{k} \cdot \frac{n-1}{k-1} \cdots \frac{n-i}{k-i} \cdots \frac{n-k+1}{1} = \frac{n!}{(n-k)!k!} = \binom{n}{k}.$$

As for the other direction, by (iii), we have

$$\binom{n}{k} \le \frac{n^k}{k!} \le \frac{n^k}{(k/e)^k} = \left(\frac{ne}{k}\right)^k,$$

■

6.2. Occupancy Problems

Problem 6.2.1. We are throwing m balls into n bins randomly (i.e., for every ball we randomly and uniformly pick a bin from the n available bins, and place the ball in the bin picked). There are many natural questions one can ask here:

(A) What is the maximum number of balls in any bin?

(B) What is the number of bins which are empty?

(C) How many balls do we have to throw, such that all the bins are non-empty, with reasonable probability?

Let X_i be the number of balls in the ith bins, when we throw n balls into n bins (i.e., $m = n$). Clearly,

$$\mathbb{E}[X_i] = \sum_{j=1}^{n} \mathbb{P}\big[\text{The } j\text{th ball fall in } i\text{th bin}\big] = n \cdot \frac{1}{n} = 1,$$

by linearity of expectation. The probability that the first bin has exactly i balls is

$$\binom{n}{i}\left(\frac{1}{n}\right)^i\left(1-\frac{1}{n}\right)^{n-i} \le \binom{n}{i}\left(\frac{1}{n}\right)^i \le \left(\frac{ne}{i}\right)^i\left(\frac{1}{n}\right)^i = \left(\frac{e}{i}\right)^i$$

This follows by Lemma 6.1.8 (iv).

Let $C_j(k)$ be the event that the jth bin has k or more balls in it. Then,

$$\mathbb{P}\big[C_1(k)\big] \leq \sum_{i=k}^{n} \left(\frac{e}{i}\right)^i \leq \left(\frac{e}{k}\right)^k \left(1 + \frac{e}{k} + \frac{e^2}{k^2} + \ldots\right) = \left(\frac{e}{k}\right)^k \frac{1}{1 - e/k}.$$

For $k^* = c \ln n / \ln\ln n$, we have

$$\begin{aligned}\mathbb{P}\big[C_1(k^*)\big] &\leq \left(\frac{e}{k^*}\right)^{k^*} \frac{1}{1 - e/k^*} \leq 2\exp\big(k^*(1 - \ln k^*)\big) \leq 2\exp\left(-\frac{k^* \ln k^*}{2}\right) \\ &\leq 2\exp\left(-\frac{c\ln n}{2\ln\ln n} \underbrace{\ln \frac{c\ln n}{\ln\ln n}}_{\approx \ln\ln n}\right) \leq 2\exp\left(-\frac{c\ln n}{4}\right) \leq \frac{1}{n^2},\end{aligned}$$

for n sufficiently large.

Let us redo this calculation more carefully (yuk!). For $k^* = \lceil (3\ln n)/\ln\ln n \rceil$. Then,

$$\begin{aligned}\mathbb{P}\big[C_1(k^*)\big] &\leq \left(\frac{e}{k^*}\right)^{k^*} \frac{1}{1 - e/k^*} \leq 2\left(\frac{e}{(3\ln n)/\ln\ln n}\right)^{k^*} = 2\big(\exp(\overbrace{1 - \ln 3}^{<0} - \ln\ln n + \ln\ln\ln n)\big)^{k^*} \\ &\leq 2\exp\big((-\ln\ln n + \ln\ln\ln n)k^*\big) \\ &\leq 2\exp\left(-3\ln n + 6\ln n \frac{\ln\ln\ln n}{\ln\ln n}\right) \leq 2\exp(-2.5\ln n) \leq \frac{1}{n^2},\end{aligned}$$

for n large enough. We conclude, that since there are n bins and they have identical distributions that

$$\mathbb{P}[\text{any bin contains more than } k^* \text{ balls}] \leq \sum_{i=1}^{n} C_i(k^*) \leq \frac{1}{n}.$$

Theorem 6.2.2. *With probability at least $1 - 1/n$, no bin has more than $k^* = \left\lceil \frac{3\ln n}{\ln\ln n} \right\rceil$ balls in it.*

Exercise 6.2.3. Show that when throwing $m = n\ln n$ balls into n bins, with probability $1 - o(1)$, every bin has $O(\log n)$ balls.

6.2.1. The Probability of all bins to have exactly one ball

Next, we are interested in the probability that all m balls fall in distinct bins. Let X_i be the event that the ith ball fell in a distinct bin from the first $i - 1$ balls. We have:

$$\begin{aligned}\mathbb{P}\big[\cap_{i=2}^{m} X_i\big] &= \mathbb{P}[X_2] \prod_{i=3}^{m} \mathbb{P}\Big[X_i \;\Big|\; \cap_{j=2}^{i-1} X_j\Big] \leq \prod_{i=2}^{m} \left(\frac{n - i + 1}{n}\right) \leq \prod_{i=2}^{m} \left(1 - \frac{i-1}{n}\right) \\ &\leq \prod_{i=2}^{m} e^{-(i-1)/n} \leq \exp\left(-\frac{m(m-1)}{2n}\right),\end{aligned}$$

thus for $m = \left\lceil \sqrt{2n} + 1 \right\rceil$, the probability that all the m balls fall in different bins is smaller than $1/e$.

This is sometime referred to as the ***birthday paradox***. You have $m = 30$ people in the room, and you ask them for the date (day and month) of their birthday (i.e., $n = 365$). The above shows that the probability of all birthdays to be distinct is $\exp(-30 \cdot 29/730) \leq 1/e$. Namely, there is more than 50% chance for a birthday collision, a simple but counter-intuitive phenomena.

6.3. The Coupon Collector's Problem

There are n types of coupons, and at each trial one coupon is picked in random. How many trials one has to perform before picking all coupons? Let m be the number of trials performed. We would like to bound the probability that m exceeds a certain number, and we still did not pick all coupons.

Let $C_i \in \{1, \ldots, n\}$ be the coupon picked in the ith trial. The jth trial is a success, if C_j was not picked before in the first $j-1$ trials. Let X_i denote the number of trials from the ith success, till after the $(i+1)$th success. Clearly, the number of trials performed is

$$X = \sum_{i=0}^{n-1} X_i.$$

Now, the probability of X_i to succeed in a trial is $p_i = (n-i)/n$, and X_i has the geometric distribution with probability p_i. As such $\mathbb{E}[X_i] = 1/p_i$, and $\mathbb{V}[X_i] = q/p^2 = (1-p_i)/p_i^2$.

Thus,

$$\mathbb{E}[X] = \sum_{i=0}^{n-1} \mathbb{E}[X_i] = \sum_{i=0}^{n-1} \frac{n}{n-i} = nH_n = n(\ln n + \Theta(1)) = n \ln n + O(n),$$

where $H_n = \sum_{i=1}^{n} 1/i$ is the nth Harmonic number.

As for variance, using the independence of $X_0, \ldots, X_{n-1}$, we have

$$\begin{aligned}
\mathbb{V}[X] &= \sum_{i=0}^{n-1} \mathbb{V}[X_i] = \sum_{i=0}^{n-1} \frac{1-p_i}{p_i^2} = \sum_{i=0}^{n-1} \frac{1-(n-i)/n}{\left(\frac{n-i}{n}\right)^2} = \sum_{i=0}^{n-1} \frac{i/n}{\left(\frac{n-i}{n}\right)^2} = \sum_{i=0}^{n-1} \frac{i}{n}\left(\frac{n}{n-i}\right)^2 \\
&= n \sum_{i=0}^{n-1} \frac{i}{(n-i)^2} = n \sum_{i=1}^{n} \frac{n-i}{i^2} = n\left(\sum_{i=1}^{n} \frac{n}{i^2} - \sum_{i=1}^{n} \frac{1}{i}\right) = n^2 \sum_{i=1}^{n} \frac{1}{i^2} - nH_n.
\end{aligned}$$

Since, $\lim_{n\to\infty} \sum_{i=1}^{n} \frac{1}{i^2} = \pi^2/6$, we have $\lim_{n\to\infty} \dfrac{\mathbb{V}[X]}{n^2} = \dfrac{\pi^2}{6}$.

Corollary 6.3.1. *Let X be the number of rounds till we collection all n coupons. Then, $\mathbb{V}[X] \approx (\pi^2/6)n^2$ and its standard deviation is $\sigma_X \approx (\pi/\sqrt{6})n$.*

This implies a weak bound on the concentration of X, using Chebyshev inequality, we have

$$\mathbb{P}\left[X \geq n \log n + n + t \cdot n\frac{\pi}{\sqrt{6}}\right] \leq \mathbb{P}\left[\left|X - \mathbb{E}[X]\right| \geq t\sigma_X\right] \leq \frac{1}{t^2},$$

Note, that this is somewhat approximate, and hold for n sufficiently large.

6.4. Notes

The material in this note covers parts of [MR95, sections 3.1,3.2,3.6]

Chapter 7

Sampling, Estimation, and More on the Coupon's Collector Problems II

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
December 10, 2019

> There is not much talking now. A silence falls upon them all. This is no time to talk of hedges and fields, or the beauties of any country. Sadness and fear and hate, how they well up in the heart and mind, whenever one opens the pages of these messengers of doom. Cry for the broken tribe, for the law and custom that is gone. Aye, and cry aloud for the man who is dead, for the woman and children bereaved. Cry, the beloved country, these things are not yet at an end. The sun pours down on the earth, on the lovely land that man cannot enjoy. He knows only the fear of his heart.
>
> Alan Paton, Cry, the beloved country

7.1. Randomized selection – Using sampling to learn the world

7.1.1. Sampling

One of the big advantages of randomized algorithms, is that they sample the world; that is, learn how the input looks like without reading all the input. For example, consider the following problem: We are given a set of U of n objects $u_1, \ldots, u_n$. and we want to compute the number of elements of U that have some property. Assume, that one can check if this property holds, in constant time, for a single object, and let $\psi(u)$ be the function that returns 1 if the property holds for the element u. and zero otherwise. Now, let Γ be the number of objects in U that have this property. We want to reliably estimate Γ without computing the property for all the elements of U.

A natural approach, would be to pick a random sample R of m objects, $r_1, \ldots, r_m$ from U (with replacement), and compute $Y = \sum_{i=1}^{m} \psi(r_1)$. The estimate for Γ is $\beta = (n/m)Y$. It is natural to ask how far is β from the true value Γ.

Lemma 7.1.1. *Let U be a set of n elements, with Γ of them having a certain property ψ. Let R be a uniform random sample from U (with repetition) of size m, and let Y be the number of elements in R that have the property ψ, and let $Z = (n/m)Y$ be the estimate for Γ. Then, for any $t \geq 1$, we have that*

$$\mathbb{P}\left[\Gamma - t\frac{n}{2\sqrt{m}} \leq Z \leq \Gamma + t\frac{n}{2\sqrt{m}}\right] \geq 1 - \frac{1}{t^2}.$$

Similarly, we have that $\mathbb{P}\left[\mathbb{E}[Y] - t\sqrt{m}/2 \leq Y \leq \mathbb{E}[Y] + t\sqrt{m}/2\right] \geq 1 - 1/t^2$.

Proof: Let $Y_i = \psi(r_i)$ be an indicator variable that is 1 if the ith sample r_i has the property ψ. Consider the random variable $Y = \sum_i Y_i$, and the estimate $Z = (n/m)Y$ for Γ. In particular, we have $\mathbb{E}[Z] = \Gamma$.

The variable Y is a binomial distribution with probability $p = \Gamma/n$, and m samples; that is, $Y \sim \mathrm{Bin}(m, p)$. We saw in the previous lecture that, $\mathbb{E}[Y] = mp$, $\mathbb{V}[Y] = mp(1-p)$, and its standard deviation is $\sigma_Y = \sqrt{mp(1-p)} \leq \sqrt{m}/2$, as $\sqrt{p(1-p)}$ is maximized for $p = 1/2$.

By Chebychev's inequality, we have that $\mathbb{P}\left[|Y - \mathbb{E}[Y]| \geq t\sigma_Y\right] \leq 1/t^2$, Since $(n/m)\,\mathbb{E}[Y] = \mathbb{E}[Z] = \Gamma$, this implies that

$$\begin{aligned}\frac{1}{t^2} &\geq \mathbb{P}\left[\left|\frac{n}{m}Y - \frac{n}{m}\,\mathbb{E}[Y]\right| \geq \frac{n}{m}t\sigma_Y\right] = \mathbb{P}\left[|Z - \Gamma| \geq \frac{n}{m}t\sigma_Y\right] \\ &\geq \mathbb{P}\left[|Z - \Gamma| \geq \frac{n}{m}t \cdot \frac{\sqrt{m}}{2}\right] = \mathbb{P}\left[|Z - \Gamma| \geq t\frac{n}{2\sqrt{m}}\right].\end{aligned}$$

■

7.1.1.1. Inverse estimation

We are given a set $U = \{u_1, \ldots, u_n\}$ of n distinct numbers. Let $U_{\langle i \rangle}$ denote the ith smallest number in U – that is $U_{\langle i \rangle}$ is the number of ***rank*** i in U.

Lemma 7.1.2. *Given a set U of n numbers, a number k, and parameters $t \geq 1$ and m, one can compute, in $O(m \log m)$ time, two numbers $r_-, r_+ \in U$, such that:*
(A) The number of rank k in U is in the interval $\mathcal{I} = [r_-, r_+]$.
(B) There are at most $8tn/\sqrt{m}$ numbers of U in $\mathcal{I}$.
The above two properties hold with probability $\geq 1 - 3/t^2$.

Proof: (A) Compute a sample R of U in $O(m)$ time (assuming the input numbers are given in an array, say). Next sort the numbers of R in $O(n \log n)$ time. Let

$$\ell_- = \left\lfloor m\frac{k}{n} - t\sqrt{m}/2 \right\rfloor - 1 \qquad \text{and} \qquad \ell_+ = \left\lceil m\frac{k}{n} + t\sqrt{m}/2 \right\rceil + 1.$$

Set $r_- = \mathsf{R}[\ell_-]$ and $r_+ = \mathsf{R}[\ell_+]$.

Let Y be the number of elements in the sample R that are $\leq U_{\langle k \rangle}$. By Lemma 7.1.1, we have $\mathbb{P}\left[\mathbb{E}[Y] - t\sqrt{m}/2 \leq Y \leq \mathbb{E}[Y] + t\sqrt{m}/2\right] \geq 1 - 1/t^2$. In particular, if this happens, then $r_- \leq U_{\langle k \rangle} \leq r_+$.

(B) Let $g = k - t\frac{n}{\sqrt{m}} - 3\frac{n}{m}$, and let g_{R} be the number of elements in R that are smaller than $U_{\langle g \rangle}$. Arguing as above, we have that $\mathbb{P}\left[g_{\mathsf{R}} \leq \frac{g}{n}m + t\sqrt{m}/2\right] \geq 1 - 1/t^2$. Now

$$\frac{g}{n}m + t\sqrt{m}/2 = \frac{m}{n}\left(k - t\frac{n}{\sqrt{m}} - 3\frac{n}{m}\right) + t\sqrt{m}/2 = k\frac{m}{n} - t\sqrt{m} - 3 + t\sqrt{m}/2 = k\frac{m}{n} - t\sqrt{m}/2 - 3 < \ell_-.$$

This implies that the g smallest numbers in U are outside the interval $[r_-, r_+]$ with probability $\geq 1 - 1/t^2$.

Next, let $h = k + t\frac{n}{\sqrt{m}} + 3\frac{n}{m}$. A similar argument, shows that all the $n - h$ largest numbers in U are too large to be in $[r_-, r_+]$. This implies that

$$|[r_-, r_+] \cap U| \leq h - g + 1 = 6\frac{n}{m} + 2t\frac{n}{\sqrt{m}} \leq 8\frac{tn}{\sqrt{m}}.$$

■

7.1.1.2. Inverse estimation – intuition

Here we are trying to give some intuition to the proof of the previous lemma. Feel free to skip this part if you feel you already understand what is going on.

Given k, we are interested in estimating $s_k = U_{\langle k \rangle}$ quickly. So, let us take a sample R of size m. Let $\mathsf{R}_{\leq s_k}$ be the set of all the numbers in R that are $\leq s_k$. For $Y = \left|\mathsf{R}_{\leq s_k}\right|$, we have that $\mu = \mathbb{E}[Y] = m\frac{k}{n}$. Furthermore, for any $t \geq 1$, Lemma 7.1.1 implies that $\mathbb{P}\left[\mu - t\sqrt{m}/2 \leq Y \leq \mu + t\sqrt{m}/2\right] \geq 1 - 1/t^2$. In particular, with probability $\geq 1 - 1/t^2$ the number $r_- = \mathsf{R}_{\langle \ell_- \rangle}$, for $\ell_- = \left\lfloor \mu - t\sqrt{m}/2 \right\rfloor - 1$, is smaller than s_k, and similarly, the number $r_+ = \mathsf{R}_{\langle \ell_+ \rangle}$ of rank $\ell_+ = \left\lceil \mu + t\sqrt{m}/2 \right\rceil + 1$ in R is larger than s_k.

One can conceptually think about the interval $\mathcal{I}(k) = [r_-, r_+]$ as *confidence interval* – we know that $s_k \in \mathcal{I}(k)$ with probability $\geq 1 - 1/t^2$. But how heavy is this interval? Namely, how many elements are there in $\mathcal{I}(k) \cap U$?

To this end, consider the interval of ranks, *in the sample*, that might contain the kth element. By the above, this is $\mathcal{I}(k,t) = k\frac{m}{n} + \left[-t\sqrt{m}/2 - 1, t\sqrt{m}/2 + 1\right]$. In particular, consider the maximum $\nu \leq k$, such that $\mathcal{I}(\nu, t)$ and $\mathcal{I}(k,t)$ are disjoint. We have the condition that $\nu\frac{m}{n} + t\sqrt{m}/2 + 1 \leq k\frac{m}{n} - t\sqrt{m}/2 - 1 \implies \nu \leq k - t\frac{n}{\sqrt{m}} - 2\frac{n}{m}$. Let $g = k - t\frac{n}{\sqrt{m}} - 2\frac{n}{m}$ and $h = k + t\frac{n}{\sqrt{m}} + 2\frac{n}{m}$. We have that $\mathcal{I}(g,t)$, $\mathcal{I}(k,t)$ and $\mathcal{I}(h,t)$ are all disjoint with probability $\geq 1 - 3/t^2$.

To this end, let $g = k - \left\lceil 2\left(t\frac{n}{2\sqrt{m}}\right)\right\rceil$ and $h = k + \left\lceil 2\left(t\frac{n}{2\sqrt{m}}\right)\right\rceil$. It is easy to verify (using the same argumentation as above) that with probability at least $1 - 3/t^2$, the three confidence $\mathcal{I}(g), \mathcal{I}(k)$ and $\mathcal{I}(h)$ do not intersect. As such, we have $\left|\mathcal{I}(k) \cap U\right| \leq h - g \leq 4\left(t\frac{n}{2\sqrt{m}}\right)$.

7.1.2. Randomized selection

7.1.2.1. The algorithm

Given an array S of n numbers, and the rank k. The algorithm needs to compute $S_{\langle k \rangle}$. To this end, set $t = \left\lceil n^{1/8} \right\rceil$, and $m = \left\lceil n^{3/4} \right\rceil$.

Using the algorithm of Lemma 7.1.2, in $O(m \log m)$ time, we get two numbers r_- and r_+, such that $S_{\langle k \rangle} \in [r_i, r_+]$, and

$$\underbrace{|S \cap (r_i, r_+)|}_{S_m} = O(tn/\sqrt{m}) = O\left(n^{1/8} n / m^{3/8}\right) = O(n^{3/4}).$$

To this end, we break S into three sets:

(i) $S_< = \{s \in S \mid s \leq r_-\}$,
(ii) $S_m = \{s \in S \mid r_- < s < r_+\}$,
(iii) $S_> = \{s \in S \mid r_+ \leq s\}$.

This three way partition can be done using $2n$ comparisons and in linear time. We now can readily compute the rank of r_- in S (it is $|S_<|$) and the rank of r_+ in S (it is $|S_<| + |S_m| + 1$). If $\operatorname{rank}(r_-, S) > k$ or $\operatorname{rank}(r_+, S) < k$ then the algorithm failed. The other possibility for failure is that S_m is too large – (i.e., larger than $8tn/\sqrt{m} = O(n^{3/4})$. If any of these failures happened, then we rerun this algorithm from scratch.

Otherwise, the algorithm need to compute the element of rank $k - |S_<|$ in the set S_m, and this can be done in $O(|S_m| \log |S_m|) = O(n^{3/4} \log n)$ time by using sorting.

LazySelect(S, k)

`Input:` S - set of n elements, k - index of element to be output.

repeat

$\mathsf{R} \leftarrow \{$Sample with replacement of $n^{3/4}$ elements from $S\}$ $\cup \{-\infty, +\infty\}$.

Sort R.

$\ell \leftarrow \max\left(1, \lfloor kn^{-1/4} - \sqrt{n} \rfloor\right)$, $h \leftarrow \min\left(n^{3/4}, \lfloor kn^{-1/4} + \sqrt{n} \rfloor\right)$

$a \leftarrow \mathsf{R}[\ell]$, $b \leftarrow \mathsf{R}[h]$.

Compute the ranks $r_S(a)$ and $r_S(b)$ of a and b in S

`/* using` $2n$ `comparisons */`

$P \leftarrow \left\{y \in S \,\middle|\, a \le y \le b\right\}$

`/* done when computing the rank of` a `and` b `*/`

Until $(r_S(a) \le k \le r_S(b))$ **and** $\left(|P| \le 8n^{3/4} + 2\right)$

Sort P in $O(n^{3/4} \log n)$ time.

return $P[k - r_S(a) + 1]$

Figure 7.1: The **LazySelect** algorithm.

7.1.2.2. Analysis

The correctness is easy – the algorithm clearly returns the desired element. As for running time, observe that by Lemma 7.1.2, by probability $\ge 1 - 1/n^{1/4}$, we succeeded in the first try, and then the running time is $O(n + (m \log m)) = O(n)$. More generally, the probability that the algorithm failed in the first α tries to get a good interval $[r_-, r_+]$ is at most $1/n^{\alpha/4}$.

Exercise 7.1.3. Given numbers r_-, r_+, show how to compute the sets $S_<, S_m, S_>$ using (only!) $1.5n$ comparisons.

Theorem 7.1.4. *Given an array S with n numbers and a rank k, one can compute the element of rank k in S in expected linear time. Formally, the resulting algorithm performs in expectation $1.5n + O(n^{3/4} \log n)$ comparisons.*

Proof: Let X be the random variable that is the number of iteration till the interval is good. We have that X is a geometric variable with probability of success $[\ge 1 - 1/n^{1/4}$. As such, the expected number of rounds till success is $\le 1/p \le 1 + 2/n^{1/4}$. As such, the expected number of comparisons performed by the algorithm is $\mathbb{E}\left[X \cdot \left(1.5n + O(n^{3/4} \log n)\right)\right] = 1.5n + O(n^{3/4} \log n)$. ∎

7.2. Randomized selection – a more direct presentation

We are given a set S of n distinct elements, with an associated ordering. For $t \in S$, let $r_S(t)$ denote the rank of t (the smallest element in S has rank 1). Let $S_{\langle i \rangle}$ denote the ith element in the sorted list of S.

Given k, we would like to compute S_k (i.e., select the kth element). The code of **LazySelect** is depicted in Figure 7.1.

Exercise 7.2.1. Show how to compute the ranks of $r_S(a)$ and $r_S(b)$, such that the expected number of comparisons performed is $1.5n$.

Consider the element $S_{(k)}$ and where it is mapped to in the random sample R. Consider the interval of values

$$\mathrm{I}(j) = \left[R_{(\alpha(j))}, R_{(\beta(j))}\right] = \left\{R_{(k)} \;\middle|\; \alpha(j) \le k \le \beta(j)\right\},$$

where $\alpha(j) = j \cdot n^{-1/4} - \sqrt{n}$ and $\beta(j) = j \cdot n^{-1/4} + \sqrt{n}$.

Lemma 7.2.2. *For a fixed j, we have that $\mathbb{P}\left[S_{(j)} \in \mathrm{I}(j)\right] \ge 1 - 1/(4n^{1/4})$.*

Proof: There are two possible bad events: (i) $S_{(j)} < R_{\alpha(j)}$ and (ii) $R_{\beta(j)} < S_{(j)}$. Let X_i be an indicator variable which is 1 if the ith sample is smaller equal to $S_{(j)}$, otherwise 0. We have $p = \mathbb{P}[X_i]] = j/n$ and $q = 1 - j/n$. The random variable $X = \sum_{i=1}^{n^{3/4}} X_i$ is the rank of $S_{(j)}$ in the random sample. Clearly, $X \sim B\left(n^{3/4}, j/n\right)$ (i.e., X has a binomial distribution with $p = j/n$, and $n^{3/4}$ trials). As such, we have $\mathbb{E}[X] = pn^{3/4}$ and $\mathbb{V}[X] = n^{3/4}pq$.

Now, by Chebyshev inequality

$$\mathbb{P}\left[\left|X - pn^{3/4}\right| \ge t\sqrt{n^{3/4}pq}\right] \le \frac{1}{t^2}.$$

Since $pn^{3/4} = jn^{-1/4}$ and $\sqrt{n^{3/4}(j/n)(1-j/n)} \le n^{3/8}/2$, we have that the probability of $a > S_{(j)}$ or $b > S_{(j)}$ is

$$\begin{aligned}
\mathbb{P}\left[S_{(j)} < R_{\alpha(j)} \text{ or } R_{\beta(j)} < S_{(j)}\right] &= \mathbb{P}\left[X < (jn^{-1/4} - \sqrt{n}) \quad \text{or} \quad X > (jn^{-1/4} + \sqrt{n})\right] \\
&= \mathbb{P}\left[|X - jn^{-1/4}| \ge 2n^{1/8} \cdot \frac{n^{3/8}}{2}\right] \\
&\le \frac{1}{\left(2n^{1/8}\right)^2} = \frac{1}{4n^{1/4}}.
\end{aligned}$$

■

Lemma 7.2.3. **LazySelect** *succeeds with probability $\ge 1 - O(n^{-1/4})$ in the first iteration. And it performs only $2n + o(n)$ comparisons.*

Proof: By Lemma 7.2.2, we know that $S_{(k)} \in \mathrm{I}(k)$ with probability $\ge 1 - 1/(4n^{1/4})$. This in turn implies that $S_{(k)} \in P$. Thus, the only possible bad event is that the set P is too large. To this end, set $k^- = k - 3n^{3/4}$ and $k^+ = k + 3n^{3/4}$, and observe that, by definition, it holds $\mathrm{I}(k^-) \cap \mathrm{I}(k) = \emptyset$ and $\mathrm{I}(k) \cap \mathrm{I}(k^+) = \emptyset$. As such, we know by Lemma 7.2.2, that $S_{(k^-)} \in \mathrm{I}(k^-)$ and $S_{(k^+)} \in \mathrm{I}(k^+)$, and this holds with probability $\ge 1 - \frac{2}{4n^{1/4}}$. As such, the set P, which is by definition contained in the range $\mathrm{I}(k)$, has only elements that are larger than $S_{(k^-)}$ and smaller than $S_{(k^+)}$. As such, the size of P is bounded by $k^+ - k^- = 6n^{3/4}$. Thus, the algorithm succeeds in the first iteration, with probability $\ge 1 - \frac{3}{4n^{1/4}}$.

As for the number of comparisons, an iteration requires

$$O(n^{3/4}\log n) + 2n + O(n^{3/4}\log n) = 2n + o(n)$$

comparisons ■

Any deterministic selection algorithm requires $2n$ comparisons, and **LazySelect** can be changed to require only $1.5n + o(n)$ comparisons (expected).

7.3. The Coupon Collector's Problem Revisited

7.3.1. Some technical lemmas

Unfortunately, in Randomized Algorithms, many of the calculations are awful[①]. As such, one has to be dexterous in approximating such calculations. We present quickly a few of these estimates.

Lemma 7.3.1. *For $x \geq 0$, we have $1-x \leq \exp(-x)$ and $1+x \leq e^x$. Namely, for all x, we have $1+x \leq e^x$.*

Proof: For $x = 0$ we have equality. Next, computing the derivative on both sides, we have that we need to prove that $-1 \leq -\exp(-x) \iff 1 \geq \exp(-x) \iff e^x \geq 1$, which clearly holds for $x \geq 0$.

A similar argument works for the second inequality. ■

Lemma 7.3.2. *For any $y \geq 1$, and $|x| \leq 1$, we have $\left(1-x^2\right)^y \geq 1 - yx^2$.*

Proof: Observe that the inequality holds with equality for $x = 0$. So compute the derivative of x of both sides of the inequality. We need to prove that

$$y(-2x)\left(1-x^2\right)^{y-1} \geq -2yx \iff \left(1-x^2\right)^{y-1} \leq 1,$$

which holds since $1 - x^2 \leq 1$, and $y - 1 \geq 0$. ■

Lemma 7.3.3. *For any $y \geq 1$, and $|x| \leq 1$, we have $\left(1-x^2y\right)e^{xy} \leq (1+x)^y \leq e^{xy}$.*

Proof: The right side of the inequality is standard by now. As for the left side. Observe that

$$(1-x^2)e^x \leq 1+x,$$

since dividing both sides by $(1+x)e^x$, we get $1-x \leq e^{-x}$, which we know holds for any x. By Lemma 7.3.2, we have

$$\left(1-x^2y\right)e^{xy} \leq \left(1-x^2\right)^y e^{xy} = \left(\left(1-x^2\right)e^x\right)^y \leq \Big(1+x\Big)^y \leq e^{xy}.$$ ■

7.3.2. Back to the coupon collector's problem

There are n types of coupons, and at each trial one coupon is picked in random. How many trials one has to perform before picking all coupons? Let m be the number of trials performed. We would like to bound the probability that m exceeds a certain number, and we still did not pick all coupons.

In the previous lecture, we showed that

$$\mathbb{P}\left[\#\text{ of trials} \geq n\log n + n + t\cdot n\frac{\pi}{\sqrt{6}}\right] \leq \frac{1}{t^2},$$

for any t.

A stronger bound, follows from the following observation. Let Z_i^r denote the event that the ith coupon was not picked in the first r trials. Clearly,

$$\mathbb{P}\left[Z_i^r\right] = \left(1-\frac{1}{n}\right)^r \leq \exp\left(-\frac{r}{n}\right).$$

[①] "In space travel," repeated Slartibartfast, "all the numbers are awful." – Life, the Universe, and Everything Else, Douglas Adams.

Thus, for $r = \beta n \log n$, we have $\mathbb{P}\Big[Z_i^r\Big] \leq \exp\left(-\frac{\beta n \log n}{n}\right) = n^{-\beta}$. Thus,

$$\mathbb{P}\Big[X > \beta n \log n\Big] \leq \mathbb{P}\left[\bigcup_i Z_i^{\beta n \log n}\right] \leq n \cdot \mathbb{P}\Big[Z_1\Big] \leq n^{-\beta+1}.$$

Lemma 7.3.4. *Let the random variable X denote the number of trials for collecting each of the n types of coupons. Then, we have $\mathbb{P}\Big[X > n \ln n + cn\Big] \leq e^{-c}$.*

Proof: The probability we fail to pick the first type of coupon is $\alpha = (1 - 1/n)^m \leq \exp\left(-\frac{n \ln n + cn}{n}\right) = \exp(-c)/n$. As such, using the union bound, the probability we fail to pick all n types of coupons is bounded by $n\alpha = \exp(-c)$, as claimed. ∎

In the following, we show a slightly stronger bound on the probability, which is $1 - \exp(-e^{-c})$. To see that it is indeed stronger, observe that $e^{-c} \geq 1 - \exp(-e^{-c})$.

7.3.3. An asymptotically tight bound

Lemma 7.3.5. *Let $c > 0$ be a constant, $m = n \ln n + cn$ for a positive integer n. Then for any constant k, we have* $\lim_{n\to\infty} \binom{n}{k}\left(1 - \frac{k}{n}\right)^m = \frac{\exp(-ck)}{k!}$.

Proof: By Lemma 7.3.3, we have

$$\left(1 - \frac{k^2 m}{n^2}\right)\exp\left(-\frac{km}{n}\right) \leq \left(1 - \frac{k}{n}\right)^m \leq \exp\left(-\frac{km}{n}\right).$$

Observe also that $\lim_{n\to\infty}\left(1 - \frac{k^2 m}{n^2}\right) = 1$, and $\exp\left(-\frac{km}{n}\right) = n^{-k}\exp(-ck)$. Also,

$$\lim_{n\to\infty} \binom{n}{k}\frac{k!}{n^k} = \lim_{n\to\infty} \frac{n(n-1)\cdots(n-k+1)}{n^k} = 1.$$

Thus, $\lim_{n\to\infty} \binom{n}{k}\left(1 - \frac{k}{n}\right)^m = \lim_{n\to\infty} \frac{n^k}{k!}\exp\left(-\frac{km}{n}\right) = \lim_{n\to\infty} \frac{n^k}{k!} n^{-k}\exp(-ck) = \frac{\exp(-ck)}{k!}$. ∎

Theorem 7.3.6. *Let the random variable X denote the number of trials for collecting each of the n types of coupons. Then, for any constant $c \in \mathbb{R}$, and $m = n \ln n + cn$, we have $\lim_{n\to\infty} \mathbb{P}\Big[X > m\Big] = 1 - \exp\left(-e^{-c}\right)$.*

Before dwelling into the proof, observe that $1 - \exp(-e^{-c}) \approx 1 - (1 - e^{-c}) = e^{-c}$. Namely, in the limit, the upper bound of Lemma 7.3.4 is tight.

Proof: We have $\mathbb{P}\Big[X > m\Big] = \mathbb{P}\big[\cup_i Z_i^m\big]$. By inclusion-exclusion, we have

$$\mathbb{P}\left[\bigcup_i Z_i^m\right] = \sum_{i=1}^{n} (-1)^{i+1} P_i^n,$$

where $P_j^n = \sum_{1 \le i_1 < i_2 < \ldots < i_j \le n} \mathbb{P}\left[\bigcap_{v=1}^{j} Z_{i_v}^m\right]$. Let $S_k^n = \sum_{i=1}^{k} (-1)^{i+1} P_i^n$. We know that $S_{2k}^n \le \mathbb{P}\left[\bigcup_i Z_i^m\right] \le S_{2k+1}^n$.

By symmetry,

$$P_k^n = \binom{n}{k} \mathbb{P}\left[\bigcap_{v=1}^{k} Z_v^m\right] = \binom{n}{k}\left(1 - \frac{k}{n}\right)^m,$$

Thus, $P_k = \lim_{n\to\infty} P_k^n = \exp(-ck)/k!$, by Lemma 7.3.5. Thus, we have

$$S_k = \sum_{j=1}^{k} (-1)^{j+1} P_j = \sum_{j=1}^{k} (-1)^{j+1} \cdot \frac{\exp(-cj)}{j!}.$$

Observe that $\lim_{k\to\infty} S_k = 1 - \exp(-e^{-c})$ by the Taylor expansion of $\exp(x)$ (for $x = -e^{-c}$). Indeed,

$$\exp\left(x\right) = \sum_{j=0}^{\infty} \frac{x^j}{j!} = \sum_{j=0}^{\infty} \frac{(-e^{-c})^j}{j!} = 1 + \sum_{j=1}^{\infty} \frac{(-1)^j \exp(-cj)}{j!}.$$

Clearly, $\lim_{n\to\infty} S_k^n = S_k$ and $\lim_{k\to\infty} S_k = 1 - \exp(-e^{-c})$. Thus, (using fluffy math), we have

$$\lim_{n\to\infty} \mathbb{P}\left[X > m\right] = \lim_{n\to\infty} \mathbb{P}\left[\cup_{i=1}^{n} Z_i^m\right] = \lim_{n\to\infty} \lim_{k\to\infty} S_k^n = \lim_{k\to\infty} S_k = 1 - \exp(-e^{-c}).$$ ■

Chapter 8

Concentration of Random Variables – Chernoff's Inequality

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
December 10, 2019

8.1. Concentration of mass and Chernoff's inequality

8.1.1. Example: Binomial distribution

Consider the binomial distribution $\mathrm{Bin}(n, 1/2)$ for various values of n as depicted in Figure 8.1 – here we think about the value of the variable as the number of heads in flipping a fair coin n times. Clearly, as the value of n increases the probability of getting a number of heads that is significantly smaller or larger than $n/2$ is tiny. Here we are interested in quantifying exactly how far can we divert from this expected value. Specifically, if $X \sim \mathrm{Bin}(n, 1/2)$, then we would be interested in bounding the probability $\mathbb{P}[X > n/2 + \Delta]$, where $\Delta = t\sigma_X = t\sqrt{n}/2$ (i.e., we are t standard deviations away from the expectation). For $t > 2$, this probability is roughly 2^{-t}, which is what we prove here.

More surprisingly, if you look only on the middle of the distribution, it looks the same after clipping away the uninteresting tails, see Figure 8.2; that is, it looks more and more like the normal distribution. This is a universal phenomena known the ***central limit theorem*** – every sum of nicely behaved random variables behaves like the normal distribution. We unfortunately need a more precise quantification of this behavior, thus the following.

8.1.2. A restricted case of Chernoff inequality via games

8.1.2.1. Chernoff games

The game. Consider the game where a player starts with $Y_0 = 1$ dollars. At every round, the player can bet a certain amount x (fractions are fine). With probability half she loses her bet, and with probability half she gains an amount equal to her bet. The player is not allowed to go all in – because if she looses then the game is over. So it is natural to ask what her optimal betting strategy is, such that in the end of the game she has as much money as possible.

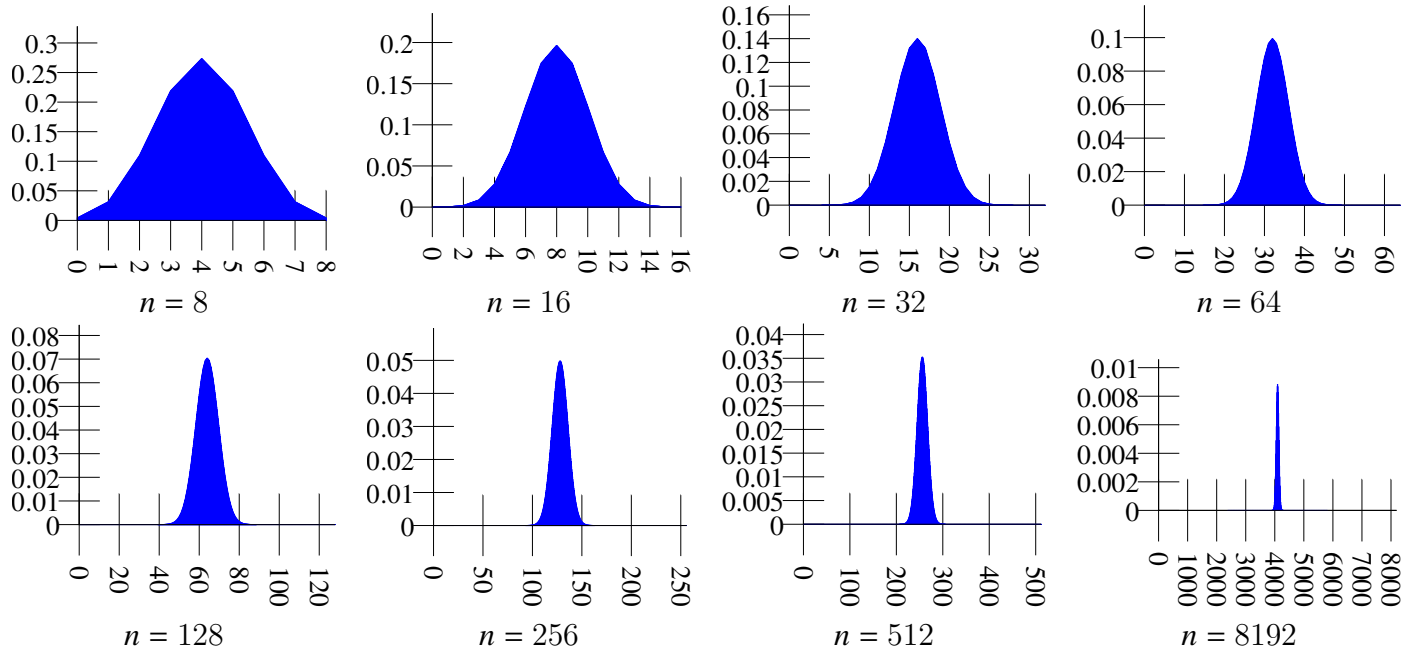


Figure 8.1: The binomial distribution for different values of n. It pretty quickly concentrates around its expectation.

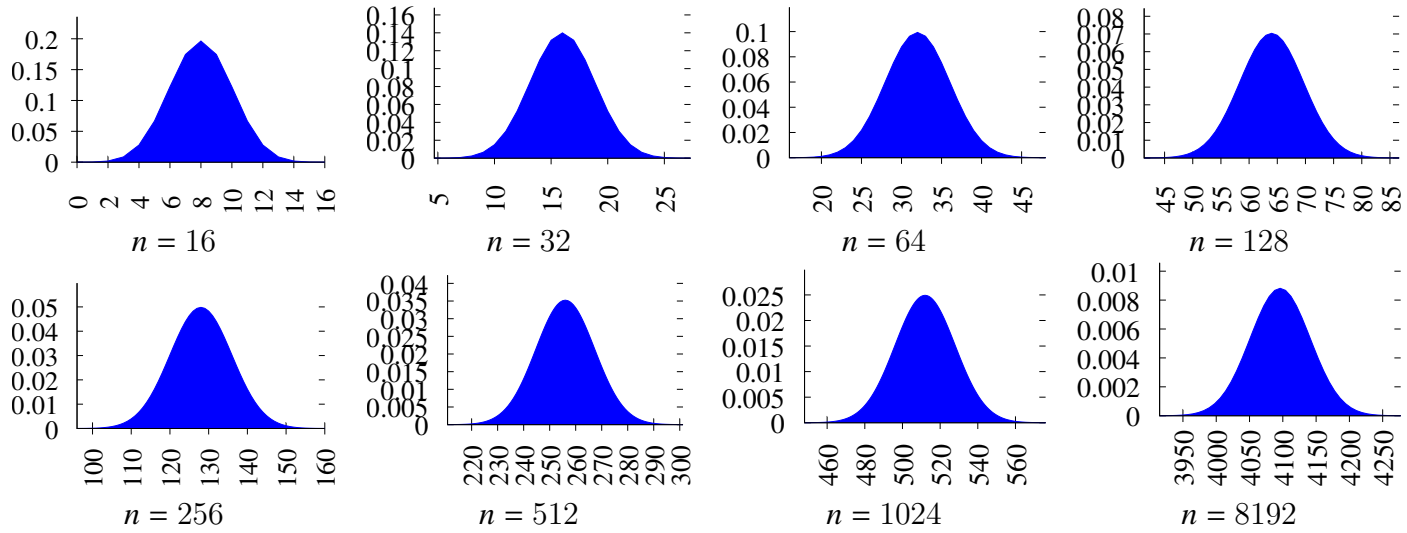


Figure 8.2: The "middle" of the binomial distribution for different values of n. It very quickly converges to the normal distribution (under appropriate rescaling and translation.

$X_i \in \{-1,+1\}$ $\mathbb{P}[X_i=-1]=\mathbb{P}[X_i=1]=1/2$	
$\mathbb{P}\big[Y \geq \Delta\big] \leq \exp\big(-\Delta^2/2n\big)$	Theorem 8.1.7
$\mathbb{P}\big[Y \leq -\Delta\big] \leq \exp\big(-\Delta^2/2n\big)$	Theorem 8.1.7

$X_i \in \{0,1\}$ $\mathbb{P}[X_i=0]=\mathbb{P}[X_i=1]=1/2$	
$\mathbb{P}\big[\lvert Y-n/2\rvert \geq \Delta\big] \leq 2\exp\big(-2\Delta^2/n\big)$	Corollary 8.1.9

$X_i \in \{0,1\}$	$\mathbb{P}[X_i=1]=p_i \qquad \mathbb{P}[X_i=0]=1-p_i$	
$\delta \geq 0$	$P=\mathbb{P}\big[Y>(1+\delta)\mu\big] < \Big(e^{\delta}\Big/(1+\delta)^{1+\delta}\Big)^{\mu}$	Theorem 8.2.1
$\delta \in (0,1)$	$P < \exp\big(-\mu\delta^2/3\big)$	Lemma 8.2.4
$\delta \in (0,4)$	$P < \exp\big(-\mu\delta^2/4\big)$	Lemma 8.2.5
$\delta \in (0,6)$	$P < \exp\big(-\mu\delta^2/5\big)$	Lemma 8.2.6
$\delta \geq 2e-1$	$P < 2^{-\mu(1+\delta)}$	Lemma 8.2.7
$\delta \geq e^2$	$P < \exp\big(-(\mu\delta/2)\ln\delta\big)$	Lemma 8.2.8
$\delta \geq 0, \varphi \in (0,1]$	$\mathbb{P}\big[Y>(1+\delta)\mu+\frac{3\ln\varphi^{-1}}{\delta^2}\big] < \varphi.$	Lemma 8.2.9
$\delta \geq 0$	$\mathbb{P}[Y<(1-\delta)\mu] < \Big(e^{-\delta}\Big/(1-\delta)^{1-\delta}\Big)^{\mu}$ $\mathbb{P}[Y<(1-\delta)\mu] < \exp\big(-\mu\delta^2/2\big)$	Theorem 8.2.3
$\Delta \geq 0$	$\mathbb{P}\big[\,Y-\mu \geq \Delta\,\big] \leq \exp\big(-2\Delta^2/n\big)$ $\mathbb{P}\big[\,Y-\mu \leq -\Delta\,\big] \leq \exp\big(-2\Delta^2/n\big).$	Corollary 8.3.5
$\tau \geq 1$	$\mathbb{P}[Y<\mu/\tau] < \exp\Big(-\big[1-\frac{1+\ln\tau}{\tau}\big]\mu\Big)$	Theorem 8.2.3

$X_i \in [0,1]$	Arbitrary independent distributions	
$\delta \in [0,1]$	$\mathbb{P}\big[Y \geq (1+\delta)\mu\big] \leq \exp\big(-\delta^2\mu/4\big)$ $\mathbb{P}\big[Y \leq (1-\delta)\mu\big] \leq \exp\big(-\delta^2\mu/2\big).$	Theorem 8.3.6
$\Delta \geq 0$	$\mathbb{P}\big[\,Y-\mu \geq \Delta\,\big] \leq \exp\big(-2\Delta^2/n\big)$ $\mathbb{P}\big[\,Y-\mu \leq -\Delta\,\big] \leq \exp\big(-2\Delta^2/n\big).$	Corollary 8.3.5

$X_i \in [a_i,b_i]$	Arbitrary independent distributions	
$\Delta \geq 0$	$\mathbb{P}\big[\lvert Y-\mu\rvert \geq \Delta\big] \leq 2\exp\Big(-\dfrac{2\Delta^2}{\sum_{i=1}^{n}(b_i-a_i)^2}\Big)$	Theorem 8.4.3

Table 8.1: Summary of Chernoff type inequalities covered. Here we have n independent random variables $X_1,\ldots,X_n$, $Y=\sum_i X_i$ and $\mu=\mathbb{E}[Y]$.

Is the game pointless? So, let Y_{i-1} be the money the player has in the end of the $(i-1)$th round, and she bets an amount $\psi_i \le Y_{i-1}$ in the ith round. As such, in the end of the ith round, she has

$$Y_i = \begin{cases} Y_{i-1} - \psi_i & \text{LOSE: probability half} \\ Y_{i-1} + \psi_i & \text{WIN: probability half} \end{cases}$$

dollars. This game, in expectation, does not change the amount of money the player has. Indeed, we have

$$\mathbb{E}\left[Y_i \,\middle|\, Y_{i-1}\right] = \frac{1}{2}(Y_{i-1} - \psi_i) + \frac{1}{2}(Y_{i-1} + \psi_i) = Y_{i-1}.$$

And as such, we have that $\mathbb{E}\left[Y_i\right] = \mathbb{E}\Big[\mathbb{E}\left[Y_i \middle| Y_{i-1}\right]\Big] = \mathbb{E}\left[Y_{i-1}\right] = \cdots = \mathbb{E}\left[Y_0\right] = 1$. In particular, $\mathbb{E}[Y_n] = 1$ – namely, on average, independent of the player strategy she is not going to make any money in this game (and she is allowed to change her bets after every round). Unless, she is lucky[①]...

What about a lucky player? The player believes she will get lucky and wants to develop a strategy to take advantage of it. Formally, she believes that she can win, say, at least $(1+\delta)/2$ fraction of her bets (instead of the predicted $1/2$) – for example, if the bets are in the stock market, she can improve her chances by doing more research on the companies she is investing in[②]. Unfortunately, the player does not know which rounds she is going to be lucky in – so she still needs to be careful.

In a search of a good strategy. Of course, there are many safe strategies the player can use, from not playing at all, to risking only a tiny fraction of her money at each round. In other words, our quest here is to find the best strategy that extracts the maximum benefit for the player out of her inherent luck.

Here, we restrict ourselves to a simple strategy – at every round, the player would bet β fraction of her money, where β is a parameter to be determined. Specifically, in the end of the ith round, the player would have

$$Y_i = \begin{cases} (1-\beta)Y_{i-1} & \text{LOSE} \\ (1+\beta)Y_{i-1} & \text{WIN.} \end{cases}$$

By our assumption, the player is going to win in at least $M = (1+\delta)n/2$ rounds. Our purpose here is to figure out what the value of β should be so that player gets as rich as possible[③]. Now, if the player is successful in $\ge M$ rounds, out of the n rounds of the game, then the amount of money the player has, in the end of the game, is

$$\begin{aligned} Y_n &\ge (1-\beta)^{n-M}(1+\beta)^M = (1-\beta)^{n/2-(\delta/2)n}(1+\beta)^{n/2+(\delta/2)n} = \Big((1-\beta)(1+\beta)\Big)^{n/2-(\delta/2)n}(1+\beta)^{\delta n} \\ &= \left(1-\beta^2\right)^{n/2-(\delta/2)n}(1+\beta)^{\delta n} \ge \exp\left(-2\beta^2\right)^{n/2-(\delta/2)n}\exp(\beta/2)^{\delta n} = \exp\left(\left(-\beta^2+\beta^2\delta+\beta\delta/2\right)n\right). \end{aligned}$$

To maximize this quantity, we choose $\beta = \delta/4$ (there is a better choice, see Lemma 8.1.6, but we use this value for the simplicity of exposition). Thus, we have that $Y_n \ge \exp\left(\left(-\frac{\delta^2}{16} + \frac{\delta^3}{16} + \frac{\delta^2}{8}\right)n\right) \ge \exp\left(\frac{\delta^2}{16}n\right)$, proving the following.

[①] "I would rather have a general who was lucky than one who was good." – Napoleon Bonaparte.

[②] "I am a great believer in luck, and I find the harder I work, the more I have of it." – Thomas Jefferson.

[③] This optimal choice is known as Kelly criterion, see Remark 8.1.3.

Lemma 8.1.1. *Consider a Chernoff game with n rounds, starting with one dollar, where the player wins in $\geq (1+\delta)n/2$ of the rounds. If the player bets $\delta/4$ fraction of her current money, at all rounds, then in the end of the game the player would have at least $\exp(n\delta^2/16)$ dollars.*

Remark 8.1.2. Note, that Lemma 8.1.1 holds if the player wins any $\geq (1+\delta)n/2$ rounds. In particular, the statement does not require randomness by itself – for our application, however, it is more natural and interesting to think about the player wins as being randomly distributed.

Remark 8.1.3. Interestingly, the idea of choosing the best fraction to bet is an old and natural question arising in investments strategies, and the right fraction to use is known as ***Kelly criterion***, going back to Kelly's work from 1956 [Kel56].

8.1.2.2. Chernoff's inequality

The above implies that if a player is lucky, then she is going to become filthy rich[4]. Intuitively, this should be a pretty rare event – because if the player is rich, then (on average) many other people have to be poor. We are thus ready for the kill.

Theorem 8.1.4 (Chernoff's inequality). *Let $X_1, \ldots, X_n$ be n independent random variables, where $X_i = 0$ or $X_i = 1$ with equal probability. Then, for any $\delta \in (0, 1/2)$, we have that*

$$\mathbb{P}\Big[\sum_i X_i \geq (1+\delta)\frac{n}{2}\Big] \leq \exp\Big(-\frac{\delta^2}{16}n\Big).$$

Proof: Imagine that we are playing the Chernoff game above, with $\beta = \delta/4$, starting with 1 dollar, and let Y_i be the amount of money in the end of the ith round. Here $X_i = 1$ indicates that the player won the ith round. We have, by Lemma 8.1.1 and Markov's inequality, that

$$\mathbb{P}\Big[\sum_i X_i \geq (1+\delta)\frac{n}{2}\Big] \leq \mathbb{P}\Big[Y_n \geq \exp\Big(\frac{n\delta^2}{16}\Big)\Big] \leq \frac{\mathbb{E}[Y_n]}{\exp(n\delta^2/16)} = \frac{1}{\exp(n\delta^2/16)} = \exp\Big(-\frac{\delta^2}{16}n\Big). \qquad \blacksquare$$

This is crazy – so intuition maybe? If the player is $(1+\delta)/2$-lucky then she can make a lot of money; specifically, at least $f(\delta) = \exp(n\delta^2/16)$ dollars by the end of the game. Namely, beating the odds has significant monetary value, and this value grows quickly with δ. Since we are in a "zero-sum" game settings, this event should be very rare indeed. Under this interpretation, of course, the player needs to know in advance the value of δ – so imagine that she guesses it somehow in advance, or she plays the game in parallel with all the possible values of δ, and she settles on the instance that maximizes her profit.

Can one do better? No, not really. Chernoff inequality is tight (this is a challenging homework exercise) up to the constant in the exponent. The best bound I know for this version of the inequality has $1/2$ instead of $1/16$ in the exponent. Note, however, that no real effort was taken to optimize the constants – this is not the purpose of this write-up.

[4] Not that there is anything wrong with that – many of my friends are filthy,

8.1.2.3. Some low level boring calculations

Above, we used the following well known facts.

Lemma 8.1.5. *(A) Markov's inequality. For any positive random variable X and $t > 0$, we have $\mathbb{P}[X \geq t] \leq \mathbb{E}[X]/t$. (B) For any two random variables X and Y, we have that $\mathbb{E}\Big[X\Big] = \mathbb{E}\Big[\mathbb{E}\big[X\big|Y\big]\Big]$. (C) For $x \in (0,1)$, $1 + x \geq e^{x/2}$. (D) For $x \in (0, 1/2)$, $1 - x \geq e^{-2x}$.*

Lemma 8.1.6. *The quantity $\exp\big(\big(-\beta^2 + \beta^2\delta + \beta\delta/2\big)n\big)$ is maximal for $\beta = \frac{\delta}{4(1-\delta)}$.*

Proof: We have to maximize $f(\beta) = -\beta^2 + \beta^2\delta + \beta\delta/2$ by choosing the correct value of β (as a function of δ, naturally). $f'(\beta) = -2\beta + 2\beta\delta + \delta/2 = 0 \iff 2(\delta - 1)\beta = -\delta/2 \iff \beta = \frac{\delta}{4(1-\delta)}$. ■

8.1.3. A proof for $-1/+1$ case

Theorem 8.1.7. *Let $X_1, \ldots, X_n$ be n independent random variables, such that $\mathbb{P}[X_i = 1] = \mathbb{P}[X_i = -1] = \frac{1}{2}$, for $i = 1, \ldots, n$. Let $Y = \sum_{i=1}^{n} X_i$. Then, for any $\Delta > 0$, we have*

$$\mathbb{P}\Big[Y \geq \Delta\Big] \leq \exp\big(-\Delta^2/2n\big).$$

Proof: Clearly, for an arbitrary t, to specified shortly, we have

$$\mathbb{P}[Y \geq \Delta] = \mathbb{P}[\exp(tY) \geq \exp(t\Delta)] \leq \frac{\mathbb{E}[\exp(tY)]}{\exp(t\Delta)},$$

the first part follows by the fact that $\exp(\cdot)$ preserve ordering, and the second part follows by the Markov inequality.

Observe that

$$\begin{aligned}
\mathbb{E}[\exp(tX_i)] &= \frac{1}{2}e^{t} + \frac{1}{2}e^{-t} = \frac{e^t + e^{-t}}{2} \\
&= \frac{1}{2}\left(1 + \frac{t}{1!} + \frac{t^2}{2!} + \frac{t^3}{3!} + \cdots\right) \\
&+ \frac{1}{2}\left(1 - \frac{t}{1!} + \frac{t^2}{2!} - \frac{t^3}{3!} + \cdots\right) \\
&= \left(1 + \quad + \frac{t^2}{2!} + \quad + \cdots + \frac{t^{2k}}{(2k)!} + \cdots\right),
\end{aligned}$$

by the Taylor expansion of $\exp(\cdot)$. Note, that $(2k)! \geq (k!)2^k$, and thus

$$\mathbb{E}[\exp(tX_i)] = \sum_{i=0}^{\infty} \frac{t^{2i}}{(2i)!} \leq \sum_{i=0}^{\infty} \frac{t^{2i}}{2^i(i!)} = \sum_{i=0}^{\infty} \frac{1}{i!}\left(\frac{t^2}{2}\right)^i = \exp\big(t^2/2\big),$$

again, by the Taylor expansion of $\exp(\cdot)$. Next, by the independence of the X_is, we have

$$\mathbb{E}[\exp(tY)] = \mathbb{E}\left[\exp\left(\sum_i tX_i\right)\right] = \mathbb{E}\left[\prod_i \exp(tX_i)\right] = \prod_{i=1}^{n} \mathbb{E}[\exp(tX_i)] \leq \prod_{i=1}^{n} e^{t^2/2} = e^{nt^2/2}.$$

We have $\mathbb{P}[Y \geq \Delta] \leq \dfrac{\exp\left(nt^2/2\right)}{\exp(t\Delta)} = \exp\left(nt^2/2 - t\Delta\right)$.

Next, by minimizing the above quantity for t, we set $t = \Delta/n$. We conclude,

$$\mathbb{P}[Y \geq \Delta] \leq \exp\left(\frac{n}{2}\left(\frac{\Delta}{n}\right)^2 - \frac{\Delta}{n}\Delta\right) = \exp\left(-\frac{\Delta^2}{2n}\right). \qquad \blacksquare$$

By the symmetry of Y, we get the following:

Corollary 8.1.8. *Let $X_1, \ldots, X_n$ be n independent random variables, such that $\mathbb{P}[X_i = 1] = \mathbb{P}[X_i = -1] = \frac{1}{2}$, for $i = 1, \ldots, n$. Let $Y = \sum_{i=1}^n X_i$. Then, for any $\Delta > 0$, we have $\mathbb{P}[|Y| \geq \Delta] \leq 2\exp\left(-\Delta^2/2n\right)$.*

Corollary 8.1.9. *Let $X_1, \ldots, X_n$ be n independent* coin flips*, such that $\mathbb{P}[X_i = 0] = \mathbb{P}[X_i = 1] = \frac{1}{2}$, for $i = 1, \ldots, n$. Let $Y = \sum_{i=1}^n X_i$. Then, for any $\Delta > 0$, we have $\mathbb{P}[|Y - n/2| \geq \Delta] \leq 2\exp\left(-2\Delta^2/n\right)$.*

Remark 8.1.10. Before going any further, it is might be instrumental to understand what this inequalities imply. Consider then case where X_i is either zero or one with probability half. In this case $\mu = \mathbb{E}[Y] = n/2$. Set $\delta = t\sqrt{n}$ ($\sqrt{\mu}$ is approximately the standard deviation of X if $p_i = 1/2$). We have by

$$\mathbb{P}\left[\left|Y - \frac{n}{2}\right| \geq \Delta\right] \leq 2\exp\left(-2\Delta^2/n\right) = 2\exp\left(-2(t\sqrt{n})^2/n\right) = 2\exp\left(-2t^2\right).$$

Thus, Chernoff inequality implies exponential decay (i.e., $\leq 2^{-t}$) with t standard deviations, instead of just polynomial (i.e., $\leq 1/t^2$) by the Chebychev's inequality.

8.2. The Chernoff Bound — General Case

Here we present the Chernoff bound in a more general settings.

Theorem 8.2.1. *Let $X_1, \ldots, X_n$ be n independent variables, where $\mathbb{P}\left[X_i = 1\right] = p_i$ and $\mathbb{P}\left[X_i = 0\right] = q_i = 1 - p_i$, for all i. Let $X = \sum_{i=1}^b X_i$. $\mu = \mathbb{E}\left[X\right] = \sum_i p_i$. For any $\delta > 0$, we have*

$$\mathbb{P}\left[X > (1+\delta)\mu\right] < \left(e^{\delta} \Big/ (1+\delta)^{1+\delta}\right)^{\mu}.$$

Proof: We have $\mathbb{P}[X > (1+\delta)\mu] = \mathbb{P}\left[e^{tX} > e^{t(1+\delta)\mu}\right]$. By the Markov inequality, we have:

$$\mathbb{P}\left[X > (1+\delta)\mu\right] < \frac{\mathbb{E}\left[e^{tX}\right]}{e^{t(1+\delta)\mu}}$$

On the other hand,

$$\mathbb{E}\left[e^{tX}\right] = \mathbb{E}\left[e^{t(X_1 + X_2 \ldots + X_n)}\right] = \mathbb{E}\left[e^{tX_1}\right] \cdots \mathbb{E}\left[e^{tX_n}\right].$$

Namely,

$$\mathbb{P}[X > (1+\delta)\mu] < \frac{\prod_{i=1}^n \mathbb{E}\left[e^{tX_i}\right]}{e^{t(1+\delta)\mu}} = \frac{\prod_{i=1}^n \left((1-p_i)e^0 + p_i e^t\right)}{e^{t(1+\delta)\mu}} = \frac{\prod_{i=1}^n \left(1 + p_i(e^t - 1)\right)}{e^{t(1+\delta)\mu}}.$$

Let $y = p_i(e^t - 1)$. We know that $1 + y < e^y$ (since $y > 0$). Thus,

$$\begin{aligned}\mathbb{P}[X > (1+\delta)\mu] &< \frac{\prod_{i=1}^n \exp(p_i(e^t-1))}{e^{t(1+\delta)\mu}} = \frac{\exp\left(\sum_{i=1}^n p_i(e^t-1)\right)}{e^{t(1+\delta)\mu}} \\ &= \frac{\exp\left((e^t-1)\sum_{i=1}^n p_i\right)}{e^{t(1+\delta)\mu}} = \frac{\exp\left((e^t-1)\mu\right)}{e^{t(1+\delta)\mu}} = \left(\frac{\exp\left(e^t-1\right)}{e^{t(1+\delta)}}\right)^{\mu} \\ &= \left(\frac{\exp(\delta)}{(1+\delta)^{(1+\delta)}}\right)^{\mu},\end{aligned}$$

if we set $t = \log(1+\delta)$. ∎

8.2.1. The lower tail

We need the following low level lemma.

Lemma 8.2.2. *For $x \in [0,1)$, we have $(1-x)^{1-x} \geq \exp(-x + x^2/2)$.*

Proof: For $x \in [0,1)$, we have, by the Taylor expansion, that $\ln(1-x) = -\sum_{i=1}^{\infty}(x^i/i)$. As such, we have

$$(1-x)\ln(1-x) = -(1-x)\sum_{i=1}^{\infty}\frac{x^i}{i} = -\sum_{i=1}^{\infty}\frac{x^i}{i} + \sum_{i=1}^{\infty}\frac{x^{i+1}}{i} = -x + \sum_{i=2}^{\infty}\left(\frac{x^i}{i-1} - \frac{x^i}{i}\right) = -x + \sum_{i=2}^{\infty}\frac{x^i}{i(i-1)}.$$

This implies that $(1-x)\ln(1-x) \geq -x + x^2/2$, which implies the claim by exponentiation. ∎

Theorem 8.2.3. *Let $X_1, \ldots, X_n$ be n independent random variables, where $\mathbb{P}\big[X_i = 1\big] = p_i$, $\mathbb{P}\big[X_i = 0\big] = q_i = 1 - p_i$, for all i. For $X = \sum_{i=1}^n X_i$, its expectation is $\mu = \mathbb{E}\big[X\big] = \sum_i p_i$. We have that*

$$\mathbb{P}\big[X < (1-\delta)\mu\big] < \left[\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right]^{\mu} \qquad \textit{or alternatively} \qquad \mathbb{P}\big[X < (1-\delta)\mu\big] < \exp\big(-\mu\delta^2/2\big).$$

For any positive $\tau > 1$, we have that $\mathbb{P}\big[X < \mu/\tau\big] \leq \exp\left(-\left(1 - \frac{1+\ln\tau}{\tau}\right)\mu\right)$.

Proof: We follow the same proof template seen already. For $t = -\ln(1-\delta) > 0$, we have $\mathbb{E}[\exp(-tX_i)] = (1-p_i)e^0 + p_i e^{-t} = 1 - p_i + p_i(1-\delta) = 1 - p_i\delta \leq \exp(-p_i\delta)$. As such, we have

$$\begin{aligned}\mathbb{P}\big[X < (1-\delta)\mu\big] = \mathbb{P}\big[-X > -(1-\delta)\mu\big] &= \mathbb{P}\big[\exp(-tX) > \exp(-t(1-\delta)\mu)\big] \leq \frac{\prod_{i=1}^n \mathbb{E}[\exp(-tX_i)]}{\exp(-t(1-\delta)\mu)} \\ &\leq \frac{\exp\left(-\sum_{i=1}^n p_i\delta\right)}{\exp(-t(1-\delta)\mu)} = \left[\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right]^{\mu}.\end{aligned}$$

The alternative simplified form, follows readily from Lemma 8.2.2, since

$$\mathbb{P}\big[X < (1-\delta)\mu\big] \leq \left[\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right]^{\mu} \leq \left[\frac{e^{-\delta}}{\exp(-\delta + \delta^2/2)}\right]^{\mu} \leq \exp(-\mu\delta^2/2).$$

For the last inequality, set $\delta = 1 - 1/\tau$, and observe that

$$\mathbb{P}\big[X < (1-\delta)\mu\big] \leq \left[\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right]^{\mu} = \left[\frac{\exp(-1+1/\tau)}{(1/\tau)^{1/\tau}}\right]^{\mu} = \exp\left(-\left(1 - \frac{1+\ln\tau}{\tau}\right)\mu\right).$$

∎

8.2.2. A more convenient form of Chernoff's inequality

Lemma 8.2.4. *Let $X_1, \ldots, X_n$ be n independent Bernoulli trials, where $\mathbb{P}[X_i = 1] = p_i$, and $\mathbb{P}[X_i = 0] = 1 - p_i$, for $i = 1, \ldots, n$. Let $X = \sum_{i=1}^{b} X_i$, and $\mu = \mathbb{E}[X] = \sum_i p_i$. For $\delta, \in (0, 1)$, we have*

$$\mathbb{P}[X > (1+\delta)\mu] < \exp(-\mu\delta^2/3).$$

Proof: By Theorem 8.2.1, it is sufficient to prove, for $\delta \in [0, 1]$, that

$$\left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu} \leq \exp\left(-\frac{\mu\delta^2}{c}\right) \iff \mu\big(\delta - (1+\delta)\ln(1+\delta)\big) \leq -\mu\delta^2/c$$

$$\iff f(\delta) = \delta^2/c + \delta - (1+\delta)\ln(1+\delta) \leq 0.$$

We have

$$f'(\delta) = 2\delta/c - \ln(1+\delta). \qquad \text{and} \qquad f''(\delta) = 2/c - \frac{1}{1+\delta}.$$

For $c = 3$, we have $f''(\delta) \leq 0$ for $\delta \in [0, 1/2]$, and $f''(\delta) \geq 0$ for $\delta \in [1/2, 1]$. Namely, $f'(\delta)$ achieves its maximum either at 0 or 1. As $f'(0) = 0$ and $f'(1) = 2/3 - \ln 2 \approx -0.02 < 0$, we conclude that $f'(\delta) \leq 0$. Namely, f is a monotonically decreasing function in $[0, 1]$, which implies that $f(\delta) \leq 0$, for all δ in this range, thus implying the claim. ■

Lemma 8.2.5. *Let $X_1, \ldots, X_n$ be n independent Bernoulli trials, where $\mathbb{P}[X_i = 1] = p_i$, and $\mathbb{P}[X_i = 0] = 1 - p_i$, for $i = 1, \ldots, n$. Let $X = \sum_{i=1}^{b} X_i$, and $\mu = \mathbb{E}[X] = \sum_i p_i$. For $\delta \in (0, 4)$, we have*

$$\mathbb{P}[X > (1+\delta)\mu] < \exp(-\mu\delta^2/4),$$

Proof: Lemma 8.2.4 implies a stronger bound, so we need to prove the claim only for $\delta \in (1, 4]$. Continuing as in the proof of Lemma 8.2.4, for case $c = 4$, we have to prove that

$$f(\delta) = \delta^2/4 + \delta - (1+\delta)\ln(1+\delta) \leq 0,$$

where $f''(\delta) = 1/2 - \frac{1}{1+\delta}$.

For $\delta > 1$, we have $f''(\delta) > 0$. Namely $f(\cdot)$ is convex for $\delta \geq 1$, and it achieves its maximum on the interval $[1, 4]$ on the endpoints. In particular, $f(1) \approx -0.13$, and $f(4) \approx -0.047$, which implies the claim. ■

Lemma 8.2.6. *Let $X_1, \ldots, X_n$ be n independent random variables, where $\mathbb{P}[X_i = 1] = p_i$, and $\mathbb{P}[X_i = 0] = 1 - p_i$, for $i = 1, \ldots, n$. Let $X = \sum_{i=1}^{b} X_i$, and $\mu = \mathbb{E}[X] = \sum_i p_i$. For $\delta \in (0, 6)$, we have*

$$\mathbb{P}[X > (1+\delta)\mu] < \exp(-\mu\delta^2/5),$$

Proof: Lemma 8.2.5 implies a stronger bound, so we need to prove the claim only for $\delta \in (4, 5]$. Continuing as in the proof of Lemma 8.2.4, for case $c = 5$, we have to prove that

$$f(\delta) = \delta^2/5 + \delta - (1+\delta)\ln(1+\delta) \leq 0,$$

where $f''(\delta) = 2/5 - \frac{1}{1+\delta}$. For $\delta \geq 4$, we have $f''(\delta) > 0$. Namely $f(\cdot)$ is convex for $\delta \geq 4$, and it achieves its maximum on the interval $[4, 6]$ on the endpoints. In particular, $f(4) \approx -0.84$, and $f(6) \approx -0.42$, which implies the claim. ■

Lemma 8.2.7. *Let $X_1, \ldots, X_n$ be n independent Bernoulli trials, where $\mathbb{P}[X_i = 1] = p_i$, and $\mathbb{P}[X_i = 0] = 1 - p_i$, for $i = 1, \ldots, n$. Let $X = \sum_{i=1}^{b} X_i$, and $\mu = \mathbb{E}\big[X\big] = \sum_i p_i$. For $\delta > 2e-1$, we have $\mathbb{P}\big[X > (1+\delta)\mu\big] < 2^{-\mu(1+\delta)}$.*

Proof: By Theorem 8.2.1, we have

$$\left(\frac{e}{1+\delta}\right)^{(1+\delta)\mu} \leq \left(\frac{e}{1+2e-1}\right)^{(1+\delta)\mu} \leq 2^{-(1+\delta)\mu},$$

since $\delta > 2e - 1$. ■

Lemma 8.2.8. *Let $X_1, \ldots, X_n$ be n independent Bernoulli trials, where $\mathbb{P}[X_i = 1] = p_i$, and $\mathbb{P}[X_i = 0] = 1 - p_i$, for $i = 1, \ldots, n$. Let $X = \sum_{i=1}^{b} X_i$, and $\mu = \mathbb{E}\big[X\big] = \sum_i p_i$. For $\delta > e^2$, we have $\mathbb{P}\big[X > (1+\delta)\mu\big] < \exp\Big(-\frac{\mu\delta \ln \delta}{2}\Big)$.*

Proof: Observe that

$$\mathbb{P}\Big[X > (1+\delta)\mu\Big] < \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu} = \exp\Big(\mu\delta - \mu(1+\delta)\ln(1+\delta)\Big). \tag{8.1}$$

As such, we have

$$\mathbb{P}\Big[X > (1+\delta)\mu\Big] < \exp\Big(-\mu(1+\delta)\big(\ln(1+\delta) - 1\big)\Big) \leq \exp\left(-\mu\delta \ln \frac{1+\delta}{e}\right) \leq \exp\left(-\frac{\mu\delta \ln \delta}{2}\right),$$

since for $x \geq e^2$ we have that $\dfrac{1+x}{e} \geq \sqrt{x} \iff \ln \dfrac{1+x}{e} \geq \dfrac{\ln x}{2}$. ■

8.2.2.1. Bound when the expectation is small

Lemma 8.2.9. *Let $X_1, \ldots, X_n$ be n independent Bernoulli trials, where $\mathbb{P}[X_i = 1] = p_i$, and $\mathbb{P}[X_i = 0] = 1 - p_i$, for $i = 1, \ldots, n$. Let $Y = \sum_{i=1}^{b} X_i$, and $\mu = \mathbb{E}[Y] = \sum_i p_i$. For $\delta \in (0, 1]$, and $\varphi \in (0, 1]$, we have*

$$\mathbb{P}\left[Y > (1+\delta)\mu + \frac{3\ln \varphi^{-1}}{\delta^2}\right] < \varphi.$$

Proof: Let $\xi = \delta + \frac{3\ln \varphi^{-1}}{\mu\delta^2}$. If $\xi \geq 2e - 1 \approx 4.43$, by Lemma 8.2.7, we have

$$\alpha = \mathbb{P}\left[Y > (1+\delta)\mu + \frac{3\ln \varphi^{-1}}{\delta^2}\right] = \mathbb{P}\Big[Y > (1+\xi)\mu\Big] \leq 2^{-\mu(1+\xi)} < \varphi,$$

since $-\mu(1+\xi) > -\mu\xi > \mu\frac{3\ln \varphi^{-1}}{\mu\delta^2} > \log_2 \varphi^{-1}$, since $\delta \in (0, 1]$.

If $\xi \leq 6$, then by Lemma 8.2.6, we have

$$\alpha = \mathbb{P}\big[Y > (1+\xi)\mu\big] \leq \exp\big(-\mu\xi^2/5\big) \leq \varphi,$$

since

$$-\frac{\mu}{5}\xi^2 = -\frac{\mu}{5}\left(\delta + \frac{3\ln \varphi^{-1}}{\mu\delta^2}\right)^2 > -\frac{\mu}{5}\left(2 \cdot \delta \cdot \frac{3\ln \varphi^{-1}}{\mu\delta^2}\right) = -\frac{6}{5} \cdot \frac{\ln \varphi}{\delta} > -\ln \varphi.$$ ■

Example 8.2.10. Let $X_1, \ldots, X_n$ be n independent Bernoulli trials, where $\mathbb{P}[X_i = 1] = p_i$, and $\mathbb{P}[X_i = 0] = 1 - p_i$, for $i = 1, \ldots, n$. Let $Y = \sum_{i=1}^{b} X_i$, and $\mu = \mathbb{E}[Y] = \sum_i p_i$. Assume that $\mu \leq 1/2$. Setting $\delta = 1$, We have, for $t > 6$, that

$$\mathbb{P}[Y > 1 + t] \leq \mathbb{P}\left[Y > (1+\delta)\mu + \frac{3\ln \exp(t/3)}{\delta^2}\right] \leq \exp(-t/3),$$

by Lemma 8.2.9.

8.3. A special case of Hoeffding's inequality

In this section, we prove yet another version of Chernoff inequality, where each variable is randomly picked according to its own distribution in the range $[0,1]$. We prove a more general version of this inequality in Section 8.4, but the version presented here does not follow from this generalization.

Theorem 8.3.1. *Let $X_1, \ldots, X_n \in [0,1]$ be n independent random variables, let $X = \sum_{i=1}^n X_i$, and let $\mu = \mathbb{E}[X]$. We have that $\mathbb{P}\Big[X - \mu \geq \eta\Big] \leq \left(\frac{\mu}{\mu+\eta}\right)^{\mu+\eta}\left(\frac{n-\mu}{n-\mu-\eta}\right)^{n-\mu-\eta}$.*

Proof: Let $s \geq 1$ be some arbitrary parameter. By the standard arguments, we have

$$\gamma = \mathbb{P}\Big[X \geq \mu+\eta\Big] = \mathbb{P}\Big[s^X \geq s^{\mu+\eta}\Big] \leq \frac{\mathbb{E}\big[s^X\big]}{s^{\mu+\eta}} = s^{-\mu-\eta}\prod_{i=1}^n \mathbb{E}\big[s^{X_i}\big].$$

By calculations, see Lemma 8.3.7 below, one can show that $\mathbb{E}\big[s^{X_1}\big] \leq 1 + (s-1)\,\mathbb{E}[X_i]$. As such, by the AM-GM inequality[5], we have that

$$\prod_{i=1}^n \mathbb{E}\big[s^{X_i}\big] \leq \prod_{i=1}^n \Big(1 + (s-1)\,\mathbb{E}[X_i]\Big) \leq \left(\frac{1}{n}\sum_{i=1}^n \Big(1 + (s-1)\,\mathbb{E}[X_i]\Big)\right)^n = \left(1 + (s-1)\frac{\mu}{n}\right)^n.$$

Setting $s = \dfrac{(\mu+\eta)(n-\mu)}{\mu(n-\mu-\eta)} = \dfrac{\mu n - \mu^2 + \eta n - \eta\mu}{\mu n - \mu^2 - \eta\mu}$ we have that

$$1 + (s-1)\frac{\mu}{n} = 1 + \frac{\eta n}{\mu n - \mu^2 - \eta\mu}\cdot\frac{\mu}{n} = 1 + \frac{\eta}{n-\mu-\eta} = \frac{n-\mu}{n-\mu-\eta}.$$

As such, we have that

$$\gamma \leq s^{-\mu-\eta}\prod_{i=1}^n \mathbb{E}\big[s^{X_i}\big] = \left(\frac{\mu(n-\mu-\eta)}{(\mu+\eta)(n-\mu)}\right)^{\mu+\eta}\left(\frac{n-\mu}{n-\mu-\eta}\right)^n = \left(\frac{\mu}{(\mu+\eta)}\right)^{\mu+\eta}\left(\frac{n-\mu}{n-\mu-\eta}\right)^{n-\mu-\eta}. \qquad \blacksquare$$

Remark 8.3.2. Setting $s = (\mu+\eta)/\mu$ in the proof of Theorem 8.3.1, we have

$$\mathbb{P}\Big[X - \mu \geq \eta\Big] \leq \left(\tfrac{\mu}{\mu+\eta}\right)^{\mu+\eta}\left(1 + \left(\tfrac{\mu+\eta}{\mu} - 1\right)\tfrac{\mu}{n}\right)^n = \left(\tfrac{\mu}{\mu+\eta}\right)^{\mu+\eta}\left(1 + \tfrac{\eta}{n}\right)^n.$$

Corollary 8.3.3. *Let $X_1, \ldots, X_n \in [0,1]$ be n independent random variables, let $\overline{X} = \sum_{i=1}^n X_i/n$, $p = \mathbb{E}\Big[\overline{X}\Big] = \mu/n$ and $q = 1-p$. Then, we have that $\mathbb{P}\Big[\,\overline{X} - p \geq t\,\Big] \leq \exp\big(n f(t)\big)$, for*

$$f(t) = (p+t)\ln\frac{p}{p+t} + (q-t)\ln\frac{q}{q-t}. \tag{8.2}$$

Theorem 8.3.4. *Let $X_1, \ldots, X_n \in [0,1]$ be n independent random variables, let $\overline{X} = \big(\sum_{i=1}^n X_i\big)/n$, and let $p = \mathbb{E}[X]$. We have that $\mathbb{P}\Big[\,\overline{X} - p \geq t\,\Big] \leq \exp\big(-2nt^2\big)$ and $\mathbb{P}\Big[\,\overline{X} - p \leq -t\,\Big] \leq \exp\big(-2nt^2\big)$.*

[5] The inequality between arithmetic and geometric means: $\big(\sum_{i=1}^n x_i\big)/n \geq \sqrt[n]{x_1 \cdots x_n}$.

Proof: Let $p = \mu/n$, $q = 1-p$, and let $f(t)$ be the function from Eq. (8.2), for $t \in (-p, q)$. Now, we have that

$$f'(t) = \ln\frac{p}{p+t} + (p+t)\frac{p+t}{p}\left(-\frac{p}{(p+t)^2}\right) - \ln\frac{q}{q-t} - (q-t)\frac{q-t}{q}\frac{q}{(q-t)^2} = \ln\frac{p}{p+t} - \ln\frac{q}{q-t}$$
$$= \ln\frac{p(q-t)}{q(p+t)}.$$

As for the second derivative, we have

$$f''(t) = \frac{\cancel{q(p+t)}}{p\cancel{(q-t)}} \cdot \frac{p}{q} \cdot \frac{(p+t)(-1) - (q-t)}{\cancel{(p+t)^2}} = \frac{-p-t-q+t}{(q-t)(p+t)} = -\frac{1}{(q-t)(p+t)} \leq -4.$$

Indeed, $t \in (-p, q)$ and the denominator is minimized for $t = (q-p)/2$, and as such $(q-t)(p+t) \leq \Big(2q-(q-p)\Big)\Big(2p+(q-p)\Big)/4 = (p+q)^2/4 = 1/4$.

Now, $f(0) = 0$ and $f'(0) = 0$, and by Taylor's expansion, we have that $f(t) = f(0) + f'(0)t + \dfrac{f''(x)}{2}t^2 \leq -2t^2$, where x is between 0 and t.

The first bound now readily follows from plugging this bound into Corollary 8.3.3. The second bound follows by considering the random variants $Y_i = 1 - X_i$, for all i, and plugging this into the first bound. Indeed, for $\overline{Y} = 1 - \overline{X}$, we have that $q = \mathbb{E}\Big[\overline{Y}\Big]$, and then $\overline{X} - p \leq -t \iff t \leq p - \overline{X} \iff t \leq 1 - q - (1-\overline{Y}) = \overline{Y} - q$. Thus, $\mathbb{P}\Big[\overline{X} - p \leq -t\Big] = \mathbb{P}\Big[\overline{Y} - q \geq t\Big] \leq \exp\big(-2nt^2\big)$. ∎

Corollary 8.3.5. *Let $X_1, \ldots, X_n \in [0,1]$ be n independent random variables, let $Y = \sum_{i=1}^n X_i$, and let $\mu = \mathbb{E}[X]$. For any $\Delta > 0$, we have $\mathbb{P}\big[\, Y - \mu \geq \Delta \,\big] \leq \exp\big(-2\Delta^2/n\big)$ and $\mathbb{P}\big[\, Y - \mu \leq -\Delta \,\big] \leq \exp\big(-2\Delta^2/n\big)$.*

Proof: For $\overline{X} = Y/n$, $p = \mu/n$, and $t = \Delta/n$, by Theorem 8.3.4, we have

$$\mathbb{P}\big[Y - \mu \geq \Delta\big] = \mathbb{P}\big[\overline{X} - p \geq t\big] \leq \exp\big(-2nt^2\big) = \exp\big(-2\Delta^2/n\big).$$ ∎

Theorem 8.3.6. *Let $X_1, \ldots, X_n \in [0,1]$ be n independent random variables, let $X = (\sum_{i=1}^n X_i)$, and let $\mu = \mathbb{E}[X]$. We have that $\mathbb{P}\big[X - \mu \geq \varepsilon\mu\big] \leq \exp\big(-\varepsilon^2\mu/4\big)$ and $\mathbb{P}\big[X - \mu \leq -\varepsilon\mu\big] \leq \exp\big(-\varepsilon^2\mu/2\big)$.*

Proof: Let $p = \mu/n$, and let $g(x) = f(px)$, for $x \in [0,1]$ and $xp < q$. As before, computing the derivative of g, we have

$$g'(x) = pf'(xp) = p\ln\frac{p(q-xp)}{q(p+xp)} = p\ln\frac{q-xp}{q(1+x)} \leq p\ln\frac{1}{1+x} \leq -\frac{px}{2},$$

since $(q-xp)/q$ is maximized for $x = 0$, and $\ln\frac{1}{1+x} \leq -x/2$, for $x \in [0,1]$, as can be easily verified[6]. Now, $g(0) = f(0) = 0$, and by integration, we have that $g(x) = \int_{y=0}^{x} g'(y)\mathrm{d}y \leq \int_{y=0}^{x}(-py/2)\mathrm{d}y = -px^2/4$. Now, plugging into Corollary 8.3.3, we get that the desired probability $\mathbb{P}\big[X - \mu \geq \varepsilon\mu\big]$ is

$$\mathbb{P}\Big[\overline{X} - p \geq \varepsilon p\Big] \leq \exp\big(nf(\varepsilon p)\big) = \exp\Big(ng(\varepsilon)\Big) \leq \exp\big(-pn\varepsilon^2/4\big) = \exp\big(-\mu\varepsilon^2/4\big).$$

[6] Indeed, this is equivalent to $\frac{1}{1+x} \leq e^{-x/2} \iff e^{x/2} \leq 1+x$, which readily holds for $x \in [0,1]$.

As for the other inequality, set $h(x) = g(-x) = f(-xp)$. Then

$$h'(x) = -pf'(-xp) = -p \ln \frac{p(q+xp)}{q(p-xp)} = p \ln \frac{q(1-x)}{q+xp} = p \ln \frac{q-xq}{q+xp} = p \ln\left(1 - x\frac{p+q}{q+xp}\right)$$
$$= p \ln\left(1 - x\frac{1}{q+xp}\right) \leq p \ln(1-x) \leq -px,$$

since $1-x \leq e^{-x}$. By integration, as before, we conclude that $h(x) \leq -px^2/2$. Now, plugging into Corollary 8.3.3, we get $\mathbb{P}\left[X - \mu \leq -\varepsilon\mu\right] = \mathbb{P}\left[\overline{X} - p \leq -\varepsilon p\right] \leq \exp\left(nf(-\varepsilon p)\right) \leq \exp\left(nh(\varepsilon)\right) \leq \exp\left(-np\varepsilon^2/2\right) \leq \exp\left(-\mu\varepsilon^2/2\right)$. ■

8.3.1. Some technical lemmas

Lemma 8.3.7. *Let $X \in [0,1]$ be a random variable, and let $s \geq 1$. Then $\mathbb{E}\left[s^X\right] \leq 1 + (s-1)\,\mathbb{E}[X]$.*

Proof: For the sake of simplicity of exposition, assume that X is a discrete random variable, and that there is a value $\alpha \in (0, 1/2)$, such that $\beta = \mathbb{P}[X = \alpha] > 0$. Consider the modified random variable X', such that $\mathbb{P}[X' = 0] = \mathbb{P}[X = 0] + \beta/2$, and $\mathbb{P}[X' = 2\alpha] = \mathbb{P}[X = \alpha] + \beta/2$. Clearly, $\mathbb{E}[X] = \mathbb{E}[X']$. Next, observe that $\mathbb{E}\left[s^{X'}\right] - \mathbb{E}\left[s^X\right] = (\beta/2)(s^{2\alpha} + s^0) - \beta s^\alpha \geq 0$, by the convexity of s^x. We conclude that $\mathbb{E}\left[s^X\right]$ achieves its maximum if takes only the values 0 and 1. But then, we have that $\mathbb{E}\left[s^X\right] = \mathbb{P}[X=0]s^0 + \mathbb{P}[X=1]s^1 = (1 - \mathbb{E}[X]) + \mathbb{E}[X]\, s = 1 + (s-1)\,\mathbb{E}[X]$, as claimed. ■

8.4. Hoeffding's inequality

In this section, we prove a generalization of Chernoff's inequality. The proof is considerably more tedious, and it is included here for the sake of completeness.

Lemma 8.4.1. *Let X be a random variable. If $\mathbb{E}[X] = 0$ and $a \leq X \leq b$, then for any $s > 0$, we have $\mathbb{E}\left[e^{sX}\right] \leq \exp\left(s^2(b-a)^2/8\right)$.*

Proof: Let $a \leq x \leq b$ and observe that x can be written as a convex combination of a and b. In particular, we have

$$x = \lambda a + (1-\lambda)b \qquad \text{for} \qquad \lambda = \frac{b-x}{b-a} \in [0,1]\,.$$

Since $s > 0$, the function $\exp(sx)$ is convex, and as such

$$e^{sx} \leq \frac{b-x}{b-a}e^{sa} + \frac{x-a}{b-a}e^{sb},$$

since we have that $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$ if $f(\cdot)$ is a convex function. Thus, for a random variable X, by linearity of expectation, we have

$$\mathbb{E}\left[e^{sX}\right] \leq \mathbb{E}\left[\frac{b-X}{b-a}e^{sa} + \frac{X-a}{b-a}e^{sb}\right] = \frac{b-\mathbb{E}[X]}{b-a}e^{sa} + \frac{\mathbb{E}[X]-a}{b-a}e^{sb}$$
$$= \frac{b}{b-a}e^{sa} - \frac{a}{b-a}e^{sb},$$

since $\mathbb{E}[X] = 0$.

Next, set $p = -\dfrac{a}{b-a}$ and observe that $1 - p = 1 + \dfrac{a}{b-a} = \dfrac{b}{b-a}$ and

$$-ps(b-a) = -\left(-\frac{a}{b-a}\right)s(b-a) = sa.$$

As such, we have

$$\begin{aligned}\mathbb{E}\left[e^{sX}\right] &\leq (1-p)e^{sa} + pe^{sb} = (1-p+pe^{s(b-a)})e^{sa} \\ &= (1-p+pe^{s(b-a)})e^{-ps(b-a)} \\ &= \exp\left(-ps(b-a) + \ln\left(1-p+pe^{s(b-a)}\right)\right) = \exp(-pu + \ln(1-p+pe^{u})),\end{aligned}$$

for $u = s(b-a)$. Setting

$$\phi(u) = -pu + \ln(1-p+pe^{u}),$$

we thus have $\mathbb{E}\left[e^{sX}\right] \leq \exp(\phi(u))$. To prove the claim, we will show that $\phi(u) \leq u^2/8 = s^2(b-a)^2/8$.

To see that, expand $\phi(u)$ about zero using Taylor's expansion. We have

$$\phi(u) = \phi(0) + u\phi'(0) + \frac{1}{2}u^2\phi''(\theta) \tag{8.3}$$

where $\theta \in [0, u]$, and notice that $\phi(0) = 0$. Furthermore, we have

$$\phi'(u) = -p + \frac{pe^{u}}{1-p+pe^{u}},$$

and as such $\phi'(0) = -p + \frac{p}{1-p+p} = 0$. Now,

$$\phi''(u) = \frac{(1-p+pe^{u})pe^{u} - (pe^{u})^2}{(1-p+pe^{u})^2} = \frac{(1-p)pe^{u}}{(1-p+pe^{u})^2}.$$

For any $x, y \geq 0$, we have $(x+y)^2 \geq 4xy$ as this is equivalent to $(x-y)^2 \geq 0$. Setting $x = 1-p$ and $y = pe^{u}$, we have that

$$\phi''(u) = \frac{(1-p)pe^{u}}{(1-p+pe^{u})^2} \leq \frac{(1-p)pe^{u}}{4(1-p)pe^{u}} = \frac{1}{4}.$$

Plugging this into Eq. (8.3), we get that

$$\phi(u) \leq \frac{1}{8}u^2 = \frac{1}{8}(s(b-a))^2 \quad \text{and} \quad \mathbb{E}\left[e^{sX}\right] \leq \exp(\phi(u)) \leq \exp\left(\frac{1}{8}(s(b-a))^2\right),$$

as claimed. ∎

Lemma 8.4.2. *Let X be a random variable. If $\mathbb{E}[X] = 0$ and $a \leq X \leq b$, then for any $s > 0$, we have*

$$\mathbb{P}[X > t] \leq \frac{\exp\left(\frac{s^2(b-a)^2}{8}\right)}{e^{st}}.$$

Proof: Using the same technique we used in proving Chernoff's inequality, we have that

$$\mathbb{P}[X > t] = \mathbb{P}\left[e^{sX} > e^{st}\right] \leq \frac{\mathbb{E}\left[e^{sX}\right]}{e^{st}} \leq \frac{\exp\left(\frac{s^2(b-a)^2}{8}\right)}{e^{st}}.$$

■

Theorem 8.4.3 (Hoeffding's inequality). *Let $X_1, \ldots, X_n$ be independent random variables, where $X_i \in [a_i, b_i]$, for $i = 1, \ldots, n$. Then, for the random variable $S = X_1 + \cdots + X_n$ and any $\eta > 0$, we have*

$$\mathbb{P}\left[\left|S - \mathbb{E}[S]\right| \geq \eta\right] \leq 2\exp\left(-\frac{2\eta^2}{\sum_{i=1}^n (b_i - a_i)^2}\right).$$

Proof: Let $Z_i = X_i - \mathbb{E}[X_i]$, for $i = 1, \ldots, n$. Set $Z = \sum_{i=1}^n Z_i$, and observe that

$$\mathbb{P}[Z \geq \eta] = \mathbb{P}\left[e^{sZ} \geq e^{s\eta}\right] \leq \frac{\mathbb{E}[\exp(sZ)]}{\exp(s\eta)},$$

by Markov's inequality. Arguing as in the proof of Chernoff's inequality, we have

$$\mathbb{E}[\exp(sZ)] = \mathbb{E}\left[\prod_{i=1}^n \exp(sZ_i)\right] = \prod_{i=1}^n \mathbb{E}[\exp(sZ_i)] \leq \prod_{i=1}^n \exp\left(\frac{s^2(b_i - a_i)^2}{8}\right),$$

since the Z_is are independent and by Lemma 8.4.1. This implies that

$$\mathbb{P}[Z \geq \eta] \leq \exp(-s\eta) \prod_{i=1}^n e^{s^2(b_i - a_i)^2/8} = \exp\left(\frac{s^2}{8}\sum_{i=1}^n (b_i - a_i)^2 - s\eta\right).$$

The upper bound is minimized for $s = 4\eta/\left(\sum_i (b_i - a_i)^2\right)$, implying

$$\mathbb{P}[Z \geq \eta] \leq \exp\left(-\frac{2\eta^2}{\sum (b_i - a_i)^2}\right).$$

The claim now follows by the symmetry of the upper bound (i.e., apply the same proof to $-Z$). ■

8.5. Bibliographical notes

Some of the exposition here follows more or less the exposition in [MR95]. Exercise 8.6.1 (without the hint) is from [Mat99]. McDiarmid [McD89] provides a survey of Chernoff type inequalities, and Theorem 8.3.6 and Section 8.3 is taken from there (our proof has somewhat weaker constants).

A more general treatment of such inequalities and tools is provided by Dubhashi and Panconesi [DP09].

8.6. Exercises

Exercise 8.6.1 (Chernoff inequality is tight.). Let $S = \sum_{i=1}^{n} S_i$ be a sum of n independent random variables each attaining values $+1$ and -1 with equal probability. Let $P(n, \Delta) = \mathbb{P}[S > \Delta]$. Prove that for $\Delta \leq n/C$,

$$P(n, \Delta) \geq \frac{1}{C} \exp\left(-\frac{\Delta^2}{Cn}\right),$$

where C is a suitable constant. That is, the well-known Chernoff bound $P(n, \Delta) \leq \exp(-\Delta^2/2n))$ is close to the truth.

Exercise 8.6.2 (Chernoff inequality is tight by direct calculations.). For this question use only basic argumentation – do not use Stirling's formula, Chernoff inequality or any similar "heavy" machinery.

(A) Prove that $\displaystyle\sum_{i=0}^{n-k} \binom{2n}{i} \leq \frac{n}{4k^2} 2^{2n}$.

Hint: Consider flipping a coin $2n$ times. Write down explicitly the probability of this coin to have at most $n-k$ heads, and use Chebyshev inequality.

(B) Using (A), prove that $\binom{2n}{n} \geq 2^{2n}/4\sqrt{n}$ (which is a pretty good estimate).

(C) Prove that $\displaystyle\binom{2n}{n+i+1} = \left(1 - \frac{2i+1}{n+i+1}\right)\binom{2n}{n+i}$.

(D) Prove that $\displaystyle\binom{2n}{n+i} \leq \exp\left(\frac{-i(i-1)}{2n}\right)\binom{2n}{n}$.

(E) Prove that $\displaystyle\binom{2n}{n+i} \geq \exp\left(-\frac{8i^2}{n}\right)\binom{2n}{n}$.

(F) Using the above, prove that $\displaystyle\binom{2n}{n} \leq c\frac{2^{2n}}{\sqrt{n}}$ for some constant c (I got $c = 0.824...$ but any reasonable constant will do).

(G) Using the above, prove that

$$\sum_{i=t\sqrt{n}+1}^{(t+1)\sqrt{n}} \binom{2n}{n-i} \leq c2^{2n} \exp\left(-t^2/2\right).$$

In particular, conclude that when flipping fair coin $2n$ times, the probability to get less than $n - t\sqrt{n}$ heads (for t an integer) is smaller than $c' \exp\left(-t^2/2\right)$, for some constant c'.

(H) Let X be the number of heads in $2n$ coin flips. Prove that for any integer $t > 0$ and any $\delta > 0$ sufficiently small, it holds that $\mathbb{P}[X < (1-\delta)n] \geq \exp\left(-c''\delta^2 n\right)$, where c'' is some constant. Namely, the Chernoff inequality is tight in the worst case.

Exercise 8.6.3 (Tail inequality for geometric variables). Let $X_1, \ldots, X_m$ be m independent random variables with geometric distribution with probability p (i.e., $\mathbb{P}[X_i = j] = (1-p)^{j-1}p$). Let $Y = \sum_i X_i$, and let $\mu = \mathbb{E}[Y] = m/p$. Prove that $\mathbb{P}\left[Y \geq (1+\delta)\mu\right] \leq \exp\left(-m\delta^2/8\right)$.

Chapter 9

Applications of Chernoff's Inequality

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
December 10, 2019

9.1. QuickSort is Quick

We revisit QuickSort. We remind the reader that the running time of QuickSort is proportional to the number of comparisons performed by the algorithm. Next, consider an arbitrary element u being sorted. Consider the ith level recursive subproblem that contains u, and let S_i be the set of elements in this subproblems. We consider u to be *successful* in the ith level, if $|S_{i+1}| \leq |S_i|/2$. Namely, if u is successful, then the next level in the recursion involving u would include a considerably smaller subproblem. Let X_i be the indicator variable which is 1 if u is successful.

We first observe that if QuickSort is applied to an array with n elements, then u can be successful at most $T = \lceil \lg n \rceil$ times, before the subproblem it participates in is of size one, and the recursion stops. Thus, consider the indicator variable X_i which is 1 if u is successful in the ith level, and zero otherwise. Note that the X_is are independent, and $\mathbb{P}[X_i = 1] = 1/2$.

If u participates in v levels, then we have the random variables $X_1, X_2, \ldots, X_v$. To make things simpler, we will extend this series by adding independent random variables, such that $\mathbb{P}[`]X_i = 1 = 1/2$, for $i \geq v$. Thus, we have an infinite sequence of independent random variables, that are 0/1 and get 1 with probability 1/2. The question is how many elements in the sequence we need to read, till we get T ones.

Lemma 9.1.1. *Let $X_1, X_2, \ldots$ be an infinite sequence of independent random 0/1 variables. Let M be an arbitrary parameter. Then the probability that we need to read more than $2M + 4t\sqrt{M}$ variables of this sequence till we collect M ones is at most $2\exp(-t^2)$, for $t \leq \sqrt{M}$. If $t \geq \sqrt{M}$ then this probability is at most $2\exp\left(-t\sqrt{M}\right)$.*

Proof: Consider the random variable $Y = \sum_{i=1}^{L} X_i$, where $L = 2M + 4t\sqrt{M}$. Its expectation is $L/2$, and using the Chernoff inequality, we get

$$\begin{aligned}\alpha = \mathbb{P}\left[Y \leq M\right] &\leq \mathbb{P}\left[|Y - L/2| \geq L/2 - M\right] \leq 2\exp\left(-\tfrac{2}{L}(L/2 - M)^2\right) \\ &\leq 2\exp\left(-2\left(M + 2t\sqrt{M} - M\right)^2/L\right) \leq 2\exp\left(-2\left(2t\sqrt{M}\right)^2/L\right) = 2\exp\left(-\frac{8t^2M}{L}\right),\end{aligned}$$

by Corollary 8.1.9. For $t \leq \sqrt{M}$ we have that $L = 2M + 4t\sqrt{M} \leq 8M$, as such in this case $\mathbb{P}[Y \leq M] \leq 2\exp(-t^2)$.

If $t \geq \sqrt{M}$, then $\alpha = 2\exp\left(-\frac{8t^2M}{2M+4t\sqrt{M}}\right) \leq 2\exp\left(-\frac{8t^2M}{6t\sqrt{M}}\right) \leq 2\exp\left(-t\sqrt{M}\right)$. ■

Going back to the **QuickSort** problem, we have that if we sort n elements, the probability that u will participate in more than $L = (4+c)\lceil \lg n \rceil = 2\lceil \lg n \rceil + 4c\sqrt{\lg n}\sqrt{\lg n}$, is smaller than $2\exp\left(-c\sqrt{\lg n}\sqrt{\lg n}\right) \leq 1/n^c$, by Lemma 9.1.1. There are n elements being sorted, and as such the probability that any element would participate in more than $(4+c+1)\lceil \lg n \rceil$ recursive calls is smaller than $1/n^c$.

Lemma 9.1.2. *For any $c > 0$, the probability that* **QuickSort** *performs more than $(6+c)n\lg n$, is smaller than $1/n^c$.*

9.2. How many times can the minimum change?

Let $\Pi = \pi_1 \ldots \pi_n$ be a random permutation of $\{1, \ldots, n\}$. Let $\mathcal{E}_i$ be the event that π_i is the minimum number seen so far as we read Π; that is, $\mathcal{E}_i$ is the event that $\pi_i = \min_{k=1}^{i} \pi_k$. Let X_i be the indicator variable that is one if $\mathcal{E}_i$ happens. We already seen, and it is easy to verify, that $\mathbb{E}[X_i] = 1/i$. We are interested in how many times the minimum might change[①]; that is $Z = \sum_i X_i$, and how concentrated is the distribution of Z. The following is maybe surprising.

Lemma 9.2.1. *The events $\mathcal{E}_1, \ldots, \mathcal{E}_n$ are independent (as such, the variables $X_1, \ldots, X_n$ are independent).*

Proof: Exercise. ■

Theorem 9.2.2. *Let $\Pi = \pi_1 \ldots \pi_n$ be a random permutation of $1, \ldots, n$, and let Z be the number of times, that π_i is the smallest number among $\pi_1, \ldots, \pi_i$, for $i = 1, \ldots, n$. Then, we have that for $t \geq 2e$ that $\mathbb{P}[Z > t\ln n] \leq 1/n^{t\ln 2}$, and for $t \in [1, 2e]$, we have that $\mathbb{P}[Z > t\ln n] \leq 1/n^{(t-1)^2/4}$.*

Proof: Follows readily from Chernoff's inequality, as $Z = \sum_i X_i$ is a sum of independent indicator variables, and, since by linearity of expectations, we have

$$\mu = \mathbb{E}[Z] = \sum_i \mathbb{E}[X_i] = \sum_{i=1}^{n} \frac{1}{i} \geq \int_{x=1}^{n+1} \frac{1}{x}\mathrm{d}x = \ln(n+1) \geq \ln n.$$

Next, we set $\delta = t - 1$, and use Theorem 8.2.1$_{\text{p71}}$. ■

9.3. Routing in a Parallel Computer

Let G be a graph of a network, where every node is a processor. The processor communicate by sending packets on the edges. Let $[0, \ldots, N-1]$ denote be vertices (i.e., processors) of G, where $N = 2^n$, and G is the hypercube. As such, each processes is identified with a binary string $b_1 b_2 \ldots b_n \in \{0,1\}^n$. Two nodes are connected if their binary string differs only in a single bit. Namely, G is the binary ***hypercube*** over n bits.

We want to investigate the best routing strategy for this topology of network. We assume that every processor need to send a message to a single other processor. This is represented by a permutation

[①] The answer, my friend, is blowing in the permutation.

RandomRoute($v_0, \ldots, v_{N-1}$)

// v_i: Packet at node i to be routed to node $d(i)$.

(i) Pick a *random* intermediate destination $\sigma(i)$ from $[1, \ldots, N]$. Packet v_i travels to $\sigma(i)$.

// Here random sampling is done with ***replacement***.

// Several packets might travel to the same destination.

(ii) Wait till all the packets arrive to their intermediate destination.

(iii) Packet v_i travels from $\sigma(i)$ to its destination $d(i)$.

Figure 9.1: The routing algorithm

π, and we would like to figure out how to send the messages encoded by the permutation while create minimum delay/congestion.

Specifically, in our model, every edge has a FIFO queue[2] of the packets it has to transmit. At every clock tick, one message get sent. All the processors start sending the packets in their permutation in the same time.

A routing scheme is ***oblivious*** if every node that has to forward a packet, inspect the packet, and depending only on the content of the packet decides how to forward it. That is, such a routing scheme is local in nature, and does not take into account other considerations. Oblivious routing is of course a bad idea – it ignores congestion in the network, and might insist routing things through regions of the hypercube that are "gridlocked".

Theorem 9.3.1 ([KKT91]). *For any deterministic oblivious permutation routing algorithm on a network of N nodes each of out-degree n, there is a permutation for which the routing of the permutation takes $\Omega\left(\sqrt{N/n}\right)$ units of time (i.e., ticks).*

Proof: (Sketch.) The above is implied by a nice averaging argument – construct, for every possible destination, the routing tree of all packets to this specific node. Argue that there must be many edges in this tree that are highly congested in this tree (which is NOT the permutation routing we are looking for!). Now, by averaging, there must be a single edge that is congested in "many" of these trees. Pick a source-destination pair from each one of these trees that uses this edge, and complete it into a full permutation in the natural way. Clearly, the congestion of the resulting permutation is high. For the exact details see [KKT91]. ■

How do we send a packet? We use ***bit fixing***. Namely, the packet from the i node, always go to the current adjacent node that have the first different bit as we scan the destination string $d(i)$. For example, packet from (0000) going to (1101), would pass through (1000), (1100), (1101).

The routing algorithm. We assume each edge have a FIFO queue. The routing algorithm is depicted in Figure 9.1.

9.3.1. Analysis

We analyze only step (i) in the algorithm, as (iii) follows from the same analysis. In the following, let ρ_i denote the route taken by v_i in (i).

[2] First in, first out queue. I sure hope you already knew that.

Exercise 9.3.2. Once a packet v_j that travel along a path ρ_j can not leave a path ρ_i, and then join it again later. Namely, $\rho_i \cap \rho_j$ is (maybe an empty) path.

Lemma 9.3.3. *Let the route of a message* $\mathbf{c}$ *follow the sequence of edges* $\pi = (e_1, e_2, \ldots, e_k)$. *Let* S *be the set of packets whose routes pass through at least one of* $(e_1, \ldots, e_k)$. *Then, the delay incurred by* $\mathbf{c}$ *is at most* $|S|$.

Proof: A packet in S is said to leave π at that time step at which it traverses an edge of π for the last time. If a packet is ready to follow edge e_j at time t, we define its *lag* at time t to be $t - j$. The lag of $\mathbf{c}$ is initially zero, and the delay incurred by $\mathbf{c}$ is its lag when it traverse e_k. We will show that each step at which the lag of $\mathbf{c}$ increases by one can be charged to a distinct member of S.

We argue that if the lag of $\mathbf{c}$ reaches $\ell + 1$, some packet in S leaves π with lag ℓ. When the lag of $\mathbf{c}$ increases from ℓ to $\ell + 1$, there must be at least one packet (from S) that wishes to traverse the same edge as $\mathbf{c}$ at that time step, since otherwise $\mathbf{c}$ would be permitted to traverse this edge and its lag would not increase. Thus, S contains at least one packet whose lag reach the value ℓ.

Let τ be the last time step at which any packet in S has lag ℓ. Thus there is a packet $\mathbf{d}$ ready to follow edge e_μ at τ, such that $\tau - \mu = \ell$. We argue that some packet of S leaves π at τ; this establishes the lemma since once a packet leaves π, it would never join it again and as such will never again delay $\mathbf{c}$.

Since $\mathbf{d}$ is ready to follow e_μ at τ, some packet ω (which may be $\mathbf{d}$ itself) in S traverses e_μ at time τ. Now ω leaves π at time τ; if not, some packet will follow $e_{\mu+1}$ at step $\mu + 1$ with lag still at ℓ, violating the maximality of τ. We charge to ω the increase in the lag of $\mathbf{c}$ from ℓ to $\ell + 1$; since ω leaves π, it will never be charged again. Thus, each member of S whose route intersects π is charge for at most one delay, establishing the lemma. ∎

Let H_{ij} be an indicator variable that is 1 if ρ_i and ρ_j share an edge, and 0 otherwise. The total delay for v_i is at most $\leq \sum_j H_{ij}$.

Crucially, for a fixed i, the variables $H_{i1}, \ldots, H_{iN}$ are independent. Indeed, imagine first picking the destination of v_i, and let the associated path be ρ_i. Now, pick the destinations of all the other packets in the network. Since the sampling of destinations is done with replacements, whether or not, the path of v_j intersects ρ_i or not, is independent of whether v_k intersects ρ_i. Of course, the probabilities $\mathbb{P}\big[H_{ij} = 1\big]$ and $\mathbb{P}\big[H_{ik} = 1\big]$ are probably different. Confusingly, however, $H_{11}, \ldots, H_{NN}$ are not independent. Indeed, imagine k and j being close vertices on the hypercube. If $H_{ij} = 1$ then intuitively it means that ρ_i is traveling close to the vertex v_j, and as such there is a higher probability that $H_{ik} = 1$.

Let $\rho_i = (e_1, \ldots, e_k)$, and let $T(e)$ be the number of packets (i.e., paths) that pass through e. We have that

$$\sum_{j=1}^{N} H_{ij} \leq \sum_{j=1}^{k} T(e_j) \quad \text{and thus} \quad \mathbb{E}\left[\sum_{j=1}^{N} H_{ij}\right] \leq \mathbb{E}\left[\sum_{j=1}^{k} T(e_j)\right].$$

Because of symmetry, the variables $T(e)$ have the same distribution for all the edges of G. On the other hand, the expected length of a path is $n/2$, there are N packets, and there are $Nn/2$ edges. We conclude $E[T(e)] = 1$. Thus

$$\mu = \mathbb{E}\left[\sum_{j=1}^{N} H_{ij}\right] \leq \mathbb{E}\left[\sum_{j=1}^{k} T(e_j)\right] = \mathbb{E}\Big[|\rho_i|\Big] \leq \frac{n}{2}.$$

By the Chernoff inequality, we have

$$\mathbb{P}\left[\sum_j H_{ij} > 7n\right] \leq \mathbb{P}\left[\sum_j H_{ij} > (1 + 13)\mu\right] < 2^{-13\mu} \leq 2^{-6n}.$$

Since there are $N = 2^n$ packets, we know that with probability $\leq 2^{-5n}$ all packets arrive to their temporary destination in a delay of most $7n$.

Theorem 9.3.4. *Each packet arrives to its destination in $\leq 14n$ stages, in probability at least $1 - 1/N$ (note that this is very conservative).*

9.4. Faraway Strings

Consider the Hamming distance between binary strings. It is natural to ask how many strings of length n can one have, such that any pair of them, is of Hamming distance at least t from each other. Consider two random strings, generated by picking at each bit randomly and independently. Thus, $\mathbb{E}[d_H(x, y)] = n/2$, where $d_H(x, y)$ denote the hamming distance between x and y. In particular, using the Chernoff inequality, we have that

$$\mathbb{P}[d_H(x, y) \leq n/2 - \Delta] \leq \exp\left(-2\Delta^2/n\right).$$

Next, consider generating M such string, where the value of M would be determined shortly. Clearly, the probability that any pair of strings are at distance at most $n/2 - \Delta$, is

$$\alpha \leq \binom{M}{2} \exp\left(-2\Delta^2/n\right) < M^2 \exp\left(-2\Delta^2/n\right).$$

If this probability is smaller than one, then there is some probability that all the M strings are of distance at least $n/2 - \Delta$ from each other. Namely, there exists a set of M strings such that every pair of them is far. We used here the fact that if an event has probability larger than zero, then it exists. Thus, set $\Delta = n/4$, and observe that

$$\alpha < M^2 \exp\left(-2n^2/16n\right) = M^2 \exp(-n/8).$$

Thus, for $M = \exp(n/16)$, we have that $\alpha < 1$. We conclude:

Lemma 9.4.1. *There exists a set of $\exp(n/16)$ binary strings of length n, such that any pair of them is at Hamming distance at least $n/4$ from each other.*

This is our first introduction to the beautiful technique known as the probabilistic method — we will hear more about it later in the course.

This result has also interesting interpretation in the Euclidean setting. Indeed, consider the sphere $\mathbb{S}$ of radius $\sqrt{n}/2$ centered at $(1/2, 1/2, \ldots, 1/2) \in \mathbb{R}^n$. Clearly, all the vertices of the binary hypercube $\{0, 1\}^n$ lie on this sphere. As such, let P be the set of points on $\mathbb{S}$ that exists according to Lemma 9.4.1. A pair p, q of points of P have *Euclidean* distance at least $\sqrt{d_H(p, q)} = \sqrt{n}4 = \sqrt{n}/2$ from each other. We conclude:

Lemma 9.4.2. *Consider the unit hypersphere $\mathbb{S}$ in $\mathbb{R}^n$. The sphere $\mathbb{S}$ contains a set Q of points, such that each pair of points is at (Euclidean) distance at least one from each other, and $|Q| \geq \exp(n/16)$.*

9.5. Bibliographical notes

Section 9.3 is based on Section 4.2 in [MR95]. A similar result to Theorem 9.3.4 is known for the case of the wrapped butterfly topology (which is similar to the hypercube topology but every node has a constant degree, and there is no clear symmetry). The interested reader is referred to [MU05].

9.6. Exercises

Exercise 9.6.1 (More binary strings. More!). To some extent, Lemma 9.4.1 is somewhat silly, as one can prove a better bound by direct argumentation. Indeed, for a fixed binary string x of length n, show a bound on the number of strings in the Hamming ball around x of radius $n/4$ (i.e., binary strings of distance at most $n/4$ from x). (Hint: interpret the special case of the Chernoff inequality as an inequality over binomial coefficients.)

Next, argue that the greedy algorithm which repeatedly pick a string which is in distance $\geq n/4$ from all strings picked so far, stops after picking at least $\exp(n/8)$ strings.

Chapter 10

Closest Pair

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
December 10, 2019

The events of September 8 prompted Foch to draft the later legendary signal: "My centre is giving way, my right is in retreat, situation excellent. I attack." It was probably never sent.

John Keegan, The first world war

10.1. How many times can a minimum change?

Let $a_1, \ldots, a_n$ be a set of n numbers, and let us randomly permute them into the sequence $b_1, \ldots, b_n$. Next, let $c_i = \min_{k=1}^{i} b_i$, and let X be the random variable which is the number of distinct values that appears in the sequence $c_1, \ldots, c_n$. What is the expectation of X?

Lemma 10.1.1. *In expectation, the number of times the minimum of a prefix of n randomly permuted numbers change, is $O(\log n)$. That is $\mathbb{E}[X] = O(\log n)$.*

Proof: Consider the indicator variable X_i, such that $X_i = 1$ if $c_i \neq c_{i-1}$. The probability for that is $\leq 1/i$, since this is the probability that the smallest number of $b_1, \ldots, b_i$ is b_i. (Why is this probability not simply equal to $1/i$?) As such, we have $X = \sum_i X_i$, and $\mathbb{E}[X] = \sum_i \mathbb{E}[X_i] = \sum_{i=1}^{n} \frac{1}{i} = O(\log n)$. ■

10.2. Closest Pair

Assumption 10.2.1. Throughout the discourse, we are going to assume that every hashing operation takes (worst case) constant time. This is quite a reasonable assumption when true randomness is available (using for example perfect hashing [CLRS01]). We will revisit this issue later in the course.

For a real positive number r and a point $\mathsf{p} = (x, y)$ in $\mathbb{R}^2$, define

$$\mathrm{G}_r(\mathsf{p}) := \left(\left\lfloor \frac{x}{r} \right\rfloor r \ , \ \left\lfloor \frac{y}{r} \right\rfloor r \right) \in \mathbb{R}^2.$$

The number r is the ***width*** of the ***grid*** G_r. Observe that G_r partitions the plane into square regions, which are *grid cells*. Formally, for any $i, j \in \mathbb{Z}$, the intersection of the half-planes $x \geq ri$, $x < r(i+1)$, $y \geq rj$ and $y < r(j+1)$ is a ***grid cell***. Further a ***grid cluster*** is a block of 3×3 contiguous grid cells.

For a point set P, and a parameter r, the partition of P into subsets by the grid G_r, is denoted by $\mathrm{G}_r(\mathsf{P})$. More formally, two points $\mathsf{p}, \mathsf{q} \in \mathsf{P}$ belong to the same set in the partition $\mathrm{G}_r(\mathsf{P})$, if both points are being mapped to the same grid point or equivalently belong to the same grid cell.

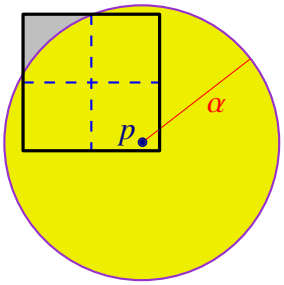
Note, that every grid cell C of G_r, has a unique ID; indeed, let $\mathsf{p} = (x, y)$ be any point in C, and consider the pair of integer numbers $\mathrm{id}_C = \mathrm{id}(\mathsf{p}) = (\lfloor x/r \rfloor, \lfloor y/r \rfloor)$. Clearly, only points inside C are going to be mapped to id_C. This is useful, as one can store a set P of points inside a grid efficiently. Indeed, given a point p, compute its $\mathrm{id}(\mathsf{p})$. We associate with each unique id a data-structure that stores all the points falling into this grid cell (of course, we do not maintain such data-structures for grid cells which are empty). For our purposes here, the grid-cell data-structure can simply be a linked list of points. So, once we computed $\mathrm{id}(\mathsf{p})$, we fetch the data structure for this cell, by using hashing. Namely, we store pointers to all those data-structures in a hash table, where each such data-structure is indexed by its unique id. Since the ids are integer numbers, we can do the hashing in constant time.

We are interested in solving the following problem.

Problem 10.2.2. Given a set P of n points in the plane, find the pair of points closest to each other. Formally, return the pair of points realizing $\mathcal{CP}(\mathsf{P}) = \min_{\mathsf{p},\mathsf{q} \in P} \|\mathsf{p} - \mathsf{q}\|$.

We need the following easy packing lemma.

Lemma 10.2.3. *Let P be a set of points contained inside a square $\square$, such that the sidelength of $\square$ is $\alpha = \mathcal{CP}(\mathsf{P})$. Then $|\mathsf{P}| \leq 4$.*



Proof: Partition $\square$ into four equal squares $\square_1, \ldots, \square_4$, and observe that each of these squares has diameter $\sqrt{2}\alpha/2 < \alpha$, and as such each can contain at most one point of P; that is, the disk of radius α centered at a point $\mathsf{p} \in \mathsf{P}$ completely covers the subsquare containing it; see the figure on the right.

Note that the set P can have four points if it is the four corners of $\square$. ■

Lemma 10.2.4. *Given a set P of n points in the plane, and a distance r, one can verify in linear time, whether or not $\mathcal{CP}(\mathsf{P}) < r$ or $\mathcal{CP}(\mathsf{P}) \geq r$.*

Proof: Indeed, store the points of P in the grid G_r. For every non-empty grid cell, we maintain a linked list of the points inside it. Thus, adding a new point p takes constant time. Indeed, compute $\mathrm{id}(p)$, check if $\mathrm{id}(p)$ already appears in the hash table, if not, create a new linked list for the cell with this ID number, and store p in it. If a data-structure already exist for $\mathrm{id}(p)$, just add p to it.

This takes $O(n)$ time. Now, if any grid cell in $\mathrm{G}_r(\mathsf{P})$ contains more than four points of P, then, by Lemma 10.2.3, it must be that the $\mathcal{CP}(\mathsf{P}) < r$.

Thus, when inserting a point p, the algorithm fetch all the points of P that were already inserted, for the cell of p, and the 8 adjacent cells. All those cells must contain at most 4 points of P (otherwise, we would already have stopped since the $\mathcal{CP}(\cdot)$ of the inserted points is smaller than r). Let S be the set of all those points, and observe that $|S| \leq 4 \cdot 9 = O(1)$. Thus, we can compute by brute force the closest point to p in S. This takes $O(1)$ time. If $\mathsf{d}(p, S) < r$, we stop and return this distance (together with the two points realizing $\mathsf{d}(p, S)$ as a proof that the distance is too short). Otherwise, we continue to the next point, where $\mathsf{d}(p, S) = \min_{s \in S} \|p - s\|$.

Overall, this takes $O(n)$ time. As for correctness, first observe that if $\mathcal{CP}(\mathsf{P}) > r$ then the algorithm would never make a mistake, since it returns '$\mathcal{CP}(\mathsf{P}) < r$' only after finding a pair of points of P with distance smaller than r. Thus, assume that p, q are the pair of points of P realizing the closest pair, and $\|p - q\| = \mathcal{CP}(\mathsf{P}) < r$. Clearly, when the later of them, say p, is being inserted, the set S would contain q, and as such the algorithm would stop and return "$\mathcal{CP}(\mathsf{P}) < r$". ■

Lemma 10.2.4 hints to a natural way to compute $\mathcal{CP}(\mathsf{P})$. Indeed, permute the points of P, in an arbitrary fashion, and let $P = \langle p_1, \ldots, p_n \rangle$. Next, let $r_i = \mathcal{CP}(\{p_1, \ldots, p_i\})$. We can check if $r_{i+1} < r_i$, by just calling the algorithm for Lemma 10.2.4 on P_{i+1} and r_i. If $r_{i+1} < r_i$, the algorithm of Lemma 10.2.4, would give us back the distance r_{i+1} (with the other point realizing this distance).

So, consider the "good" case where $r_{i+1} = r_i = r_{i-1}$. Namely, the length of the shortest pair does not change. In this case we do not need to rebuild the data structure of Lemma 10.2.4 for each point. We can just reuse it from the previous iteration. Thus, inserting a single point takes constant time as long as the closest pair (distance) does not change.

Things become bad, when $r_i < r_{i-1}$. Because then we need to rebuild the grid, and reinsert all the points of $\mathsf{P}_i = \langle p_1, \ldots, p_i \rangle$ into the new grid $\mathrm{G}_{r_i}(\mathsf{P}_i)$. This takes $O(i)$ time.

So, if the closest pair radius, in the sequence $r_1, \ldots, r_n$, changes only k times, then the running time of the algorithm would be $O(nk)$. But we can do even better!

Theorem 10.2.5. *Let P be a set of n points in the plane. One can compute the closest pair of points of P in expected linear time.*

Proof: Pick a random permutation of the points of P, and let $\langle p_1, \ldots, p_n \rangle$ be this permutation. Let $r_2 = \|p_1 - p_2\|$, and start inserting the points into the data structure of Lemma 10.2.4. In the ith iteration, if $r_i = r_{i-1}$, then this insertion takes constant time. If $r_i < r_{i-1}$, then we rebuild the grid and reinsert the points. Namely, we recompute $\mathrm{G}_{r_i}(\mathsf{P}_i)$.

To analyze the running time of this algorithm, let X_i be the indicator variable which is 1 if $r_i \neq r_{i-1}$, and 0 otherwise. Clearly, the running time is proportional to

$$R = 1 + \sum_{i=2}^{n} (1 + X_i \cdot i).$$

Thus, the expected running time is

$$\mathbb{E}[R] = 1 + \mathbb{E}\left[1 + \sum_{i=2}^{n} (1 + X_i \cdot i)\right] = n + \sum_{i=2}^{n} \big(\mathbb{E}[X_i] \cdot i\big) = n + \sum_{i=2}^{n} i \cdot \mathbb{P}[X_1 = 1],$$

by linearity of expectation and since for an indicator variable X_i, we have that $\mathbb{E}[X_i] = \mathbb{P}[X_i = 1]$.

Thus, we need to bound $\mathbb{P}[X_i = 1] = \mathbb{P}[r_i < r_{i-1}]$. To bound this quantity, fix the points of P_i, and randomly permute them. A point $\mathsf{q} \in \mathsf{P}_i$ is ***critical*** if $\mathcal{CP}(\mathsf{P}_i \setminus \{\mathsf{q}\}) > \mathcal{CP}(\mathsf{P}_i)$.

(A) If there are no critical points, then $r_{i-1} = r_i$ and then $\mathbb{P}[X_i = 1] = 0$.

(B) If there is one critical point, than $\mathbb{P}[X_i = 1] = 1/i$, as this is the probability that this critical point would be the last point in a random permutation of P_i.

(C) If there are two critical points, and let p, q be this unique pair of points of P_i realizing $\mathcal{CP}(\mathsf{P}_i)$. The quantity r_i is smaller than r_{i-1}, if either p or q are p_i. But the probability for that is $2/i$ (i.e., the probability in a random permutation of i objects, that one of two marked objects would be the last element in the permutation).

Observe, that there can not be more than two critical points. Indeed, if p and q are two points that realize the closest distance, than if there is a third critical point r, then $\mathcal{CP}(\mathsf{P}_i \setminus \{\mathsf{r}\}) = \|\mathsf{p} - \mathsf{q}\|$, and r is not critical.

We conclude that

$$\mathbb{E}[R] = n + \sum_{i=2}^{n} i \cdot \mathbb{P}[X_1 = 1] \leq n + \sum_{i=2}^{n} i \cdot \frac{2}{i} \leq 3n.$$

As such, the expected running time of this algorithm is $O(\mathbb{E}[R]) = O(n)$. ∎

Theorem 10.2.5 is a surprising result, since it implies that ***uniqueness*** (i.e., deciding if n real numbers are all distinct) can be solved in linear time. However, there is a lower bound of $\Omega(n \log n)$ on uniqueness, using the comparison tree model. This reality dysfunction, can be easily explained, once one realizes that the model of computation of Theorem 10.2.5 is considerably stronger, using hashing, randomization, and the floor function.

10.3. Bibliographical notes

The closest-pair algorithm follows Golin *et al.* [GRSS95]. This is in turn a simplification of a result of the celebrated result of Rabin [Rab76]. Smid provides a survey of such algorithms [Smi00]. A generalization of the closest pair algorithm was provided by Har-Peled and Raichel [HR15].

Surprisingly, Schönhage [Sch79] showed that assuming that the floor function is allowed, and the standard arithmetic operation can be done in constant time, then every problem in PSPACE can be solved in polynomial time. Since PSPACE includes NPC, this is bad news, as it implies that one can solve NPC problem in polynomial time (finally!). The basic idea is that one can pack huge number of bits into a single number, and the floor function enables one to read a single bit of this number. As such, a real RAM model that allows certain operations, and put no limit on the bit complexity of numbers, and assume that each operation can take constant time, is not a reasonable model of computation (but we already knew that).

Chapter 11

Backwards analysis

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
December 10, 2019

The idea of ***backwards analysis*** (or backward analysis) is a technique to analyze randomized algorithms by imagining as if it was running backwards in time, from output to input. Most of the more interesting applications of backward analysis are in Computational Geometry, but nevertheless, there are some other applications that are interesting and we survey some of them here.

11.1. How many times can the minimum change?

Let $\Pi = \pi_1 \ldots \pi_n$ be a random permutation of $\{1, \ldots, n\}$. Let $\mathcal{E}_i$ be the event that π_i is the minimum number seen so far as we read Π; that is, $\mathcal{E}_i$ is the event that $\pi_i = \min_{k=1}^{i} \pi_k$. Let X_i be the indicator variable that is one if $\mathcal{E}_i$ happens. We already seen, and it is easy to verify, that $\mathbb{E}[X_i] = 1/i$. We are interested in how many times the minimum might change[①]; that is $Z = \sum_i X_i$, and how concentrated is the distribution of Z. The following is maybe surprising.

Lemma 11.1.1. *The events $\mathcal{E}_1, \ldots, \mathcal{E}_n$ are independent (as such, variables $X_1, \ldots, X_n$ are independent).*

Proof: The trick is to think about the sampling process in a different way, and then the result readily follows. Indeed, we randomly pick a permutation of the given numbers, and set the first number to be π_n. We then, again, pick a random permutation of the remaining numbers and set the first number as the penultimate number (i.e., π_{n-1}) in the output permutation. We repeat this process till we generate the whole permutation.

Now, consider $1 \le i_1 < i_2 < \ldots < i_k \le n$, and observe that $\mathbb{P}\Big[\mathcal{E}_{i_1} \Big| \mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}\Big] = \mathbb{P}\big[\mathcal{E}_{i_1}\big]$, since by our thought experiment, $\mathcal{E}_{i_1}$ is determined after all the other variables $\mathcal{E}_{i_2}, \ldots, \mathcal{E}_{i_k}$. In particular, the variable $\mathcal{E}_{i_1}$ is inherently not effected by these events happening or not. As such, we have

$$\begin{aligned}\mathbb{P}\Big[\mathcal{E}_{i_1} \cap \mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}\Big] &= \mathbb{P}\Big[\mathcal{E}_{i_1} \Big| \mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}\Big] \mathbb{P}\big[\mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}\big] \\ &= \mathbb{P}\Big[\mathcal{E}_{i_1}\Big] \mathbb{P}\Big[\mathcal{E}_{i_2} \cap \mathcal{E}_{i_2} \cap \ldots \cap \mathcal{E}_{i_k}\Big] = \prod_{j=1}^{k} \mathbb{P}\big[\mathcal{E}_{i_j}\big] = \prod_{j=1}^{k} \frac{1}{i_j},\end{aligned}$$

by induction. ∎

[①] The answer, my friend, is blowing in the permutation.

Theorem 11.1.2. *Let $\Pi = \pi_1 \dots \pi_n$ be a random permutation of $1, \dots, n$, and let Z be the number of times, that π_i is the smallest number among $\pi_1, \dots, \pi_i$, for $i = 1, \dots, n$. Then, we have that for $t \geq 2e$ that $\mathbb{P}\Big[Z > t \ln n\Big] \leq 1/n^{t \ln 2}$, and for $t \in \Big[1, 2e\Big]$, we have that $\mathbb{P}\Big[Z > t \ln n\Big] \leq 1/n^{(t-1)^2/4}$.*

Proof: Follows readily from Chernoff's inequality, as $Z = \sum_i X_i$ is a sum of independent indicator variables, and, since by linearity of expectations, we have

$$\mu = \mathbb{E}\Big[Z\Big] = \sum_i \mathbb{E}\Big[X_i\Big] = \sum_{i=1}^{n} \frac{1}{i} \geq \int_{x=1}^{n+1} \frac{1}{x} \mathrm{d}x = \ln(n+1) \geq \ln n.$$

Next, we set $\delta = t - 1$, and use Chernoff inequality. ∎

11.2. Computing a good ordering of the vertices of a graph

We are given a $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ be an edge-weighted graph with n vertices and m edges. The task is to compute an ordering $\pi = \langle \pi_1, \dots, \pi_n \rangle$ of the vertices, and for every vertex $v \in \mathsf{V}$, the list of vertices L_v, such that $\pi_i \in Ł_v$, if π_i is the closet vertex to v in the ith prefix $\langle \pi_1, \dots, \pi_i \rangle$.

This situation can arise for example in a streaming scenario, where we install servers in a network. In the ith stage there i servers installed, and every client in the network wants to know its closest server. As we install more and more servers (ultimately, every node is going to be server), each client needs to maintain its current closest server.

The purpose is to minimize the total size of these lists $\mathcal{L} = \sum_{v \in \mathsf{V}} |L_v|$.

11.2.1. The algorithm

Take a random permutation $\pi_1, \dots, \pi_n$ of the vertices V of G. Initially, we set $\delta(v) = +\infty$, for all $v \in \mathsf{V}$.

In the ith iteration, set $\delta(\pi_i)$ to 0, and start Dijkstra from the ith vertex π_i. The Dijkstra propagates only if it improves the current distance associated with a vertex. Specifically, in the ith iteration, we update $\delta(u)$ to $\mathsf{d}_{\mathsf{G}}(\pi_i, u)$ if and only if $\mathsf{d}_{\mathsf{G}}(\pi_i, u) < \delta(u)$ before this iteration started. If $\delta(u)$ is updated, then we add π_i to L_u. Note, that this Dijkstra propagation process might visit only small portions of the graph in some iterations – since it improves the current distance only for few vertices.

11.2.2. Analysis

Lemma 11.2.1. *The above algorithm computes a permutation π, such that $\mathbb{E}\Big[|\mathcal{L}|\Big] = O(n \log n)$, and the expected running time of the algorithm is $O\Big((n \log n + m) \log n\Big)$, where $n = |V(\mathsf{G})|$ and $m = |\mathsf{E}(\mathsf{G})|$. Note, that both bounds also hold with high probability.*

Proof: Fix a vertex $v \in \mathsf{V} = \{v_1, \dots, v_n\}$. Consider the set of n numbers $\{\mathsf{d}_{\mathsf{G}}(v, v_1), \dots, \mathsf{d}_{\mathsf{G}}(v, v_n)\}$. Clearly, $\mathsf{d}_{\mathsf{G}}(v, \pi_1), \dots, \mathsf{d}_{\mathsf{G}}(v, \pi_n)$ is a random permutation of this set, and by Lemma 11.1.1 the random permutation π changes this minimum $O(\log n)$ time in expectations (and also with high probability). This readily implies that $|L_v| = O(\log n)$ both in expectations and high probability.

The more interesting claim is the running time. Consider an edge $uv \in \mathsf{E}(\mathsf{G})$, and observe that $\delta(u)$ or $\delta(v)$ changes $O(\log n)$ times. As such, an edge gets visited $O(\log n)$ times, which implies overall running time of $O(n \log^2 n + m \log n)$, as desired.

Indeed, overall there are $O(n \log n)$ changes in the value of $\delta(\cdot)$. Each such change might require one **delete-min** operation from the queue, which takes $O(\log n)$ time operation. Every edge, by the above, might trigger $O(\log n)$ **decrease-key** operations. Using Fibonacci heaps, each such operation takes $O(1)$ time. ■

11.3. Computing nets

11.3.1. Basic definitions

Definition 11.3.1. A ***metric space*** is a pair $(\mathcal{X}, \mathsf{d})$ where $\mathcal{X}$ is a set and $\mathsf{d} : \mathcal{X} \times \mathcal{X} \to [0, \infty)$ is a ***metric*** satisfying the following axioms: (i) $\mathsf{d}(x, y) = 0$ if and only if $x = y$, (ii) $\mathsf{d}(x, y) = \mathsf{d}(y, x)$, and (iii) $\mathsf{d}(x, y) + \mathsf{d}(y, z) \geq \mathsf{d}(x, z)$ (triangle inequality).

For example, $\mathbb{R}^2$ with the regular Euclidean distance is a metric space. In the following, we assume that we are given ***black-box access*** to $\mathsf{d}_{\mathcal{M}}$. Namely, given two points $\mathsf{p}, \mathsf{q} \in \mathcal{X}$, we assume that $\mathsf{d}(\mathsf{p}, \mathsf{q})$ can be computed in constant time.

Another standard example for a finite metric space is a graph G with non-negative weights $\omega(\cdot)$ defined on its edges. Let $\mathsf{d}_{\mathsf{G}}(x, y)$ denote the shortest path (under the given weights) between any $x, y \in V(\mathsf{G})$. It is easy to verify that $\mathsf{d}_{\mathsf{G}}(\cdot, \cdot)$ is a metric. In fact, any ***finite metric*** (i.e., a metric defined over a finite set) can be represented by such a weighted graph.

11.3.1.1. Nets

Definition 11.3.2. For a point set P in a metric space with a metric d, and a parameter $r > 0$, an ***r-net*** of P is a subset $C \subseteq \mathsf{P}$, such that

(i) for every $\mathsf{p}, \mathsf{q} \in C$, $\mathsf{p} \neq \mathsf{q}$, we have that $\mathsf{d}(\mathsf{p}, \mathsf{q}) \geq r$, and
(ii) for all $\mathsf{p} \in \mathsf{P}$, we have that $\min_{\mathsf{q} \in C} \mathsf{d}(\mathsf{p}, \mathsf{q}) < r$.

Intuitively, an r-net represents P in resolution r.

11.3.2. Computing an r-net in a sparse graph

Given a $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ be an edge-weighted graph with n vertices and m edges, and let $r > 0$ be a parameter. We are interested in the problem of computing an r-net for G. That is, a set of vertices of G that complies with Definition 11.3.2$_{\text{p93}}$.

11.3.2.1. The algorithm

We compute an r-net in a sparse graph using a variant of Dijkstra's algorithm with the sequence of starting vertices chosen in a random permutation.

Let π_i be the ith vertex in a random permutation π of V. For each vertex v we initialize $\delta(v)$ to $+\infty$. In the ith iteration, we test whether $\delta(\pi_i) \geq r$, and if so we do the following steps:

(A) Add π_i to the resulting net $\mathcal{N}$.
(B) Set $\delta(\pi_i)$ to zero.
(C) Perform Dijkstra's algorithm starting from π_i, modified to avoid adding a vertex u to the priority queue unless its tentative distance is smaller than the current value of $\delta(u)$. When such a vertex u is expanded, we set $\delta(u)$ to be its computed distance from π_i, and relax the edges adjacent to u in the graph.

11.3.2.2. Analysis

While the analysis here does not directly uses backward analysis, it is inspired to a large extent by such an analysis as in Section 11.2_{p92}.

Lemma 11.3.3. *The set $\mathcal{N}$ is an r-net in G.*

Proof: By the end of the algorithm, each $v \in \mathsf{V}$ has $\delta(v) < r$, for $\delta(v)$ is monotonically decreasing, and if it were larger than r when v was visited then v would have been added to the net.

An induction shows that if $\ell = \delta(v)$, for some vertex v, then the distance of v to the set $\mathcal{N}$ is at most ℓ. Indeed, for the sake of contradiction, let j be the (end of) the first iteration where this claim is false. It must be that $\pi_j \in \mathcal{N}$, and it is the nearest vertex in $\mathcal{N}$ to v. But then, consider the shortest path between π_j and v. The modified Dijkstra must have visited all the vertices on this path, thus computing $\delta(v)$ correctly at this iteration, which is a contradiction.

Finally, observe that every two points in $\mathcal{N}$ have distance $\geq r$. Indeed, when the algorithm handles vertex $v \in \mathcal{N}$, its distance from all the vertices currently in $\mathcal{N}$ is $\geq r$, implying the claim. ■

Lemma 11.3.4. *Consider an execution of the algorithm, and any vertex $v \in \mathsf{V}$. The expected number of times the algorithm updates the value of $\delta(v)$ during its execution is $O(\log n)$, and more strongly the number of updates is $O(\log n)$ with high probability.*

Proof: For simplicity of exposition, assume all distances in G are distinct. Let S_i be the set of all the vertices $x \in \mathsf{V}$, such that the following two properties both hold:

(A) $\mathsf{d}_{\mathsf{G}}(x, v) < \mathsf{d}_{\mathsf{G}}(v, \Pi_i)$, where $\Pi_i = \{\pi_1, \ldots, \pi_i\}$.

(B) If $\pi_{i+1} = x$ then $\delta(v)$ would change in the $(i+1)$th iteration.

Let $s_i = |S_i|$. Observe that $S_1 \supseteq S_2 \supseteq \cdots \supseteq S_n$, and $|S_n| = 0$.

In particular, let $\mathcal{E}_{i+1}$ be the event that $\delta(v)$ changed in iteration $(i+1)$ – we will refer to such an iteration as being ***active***. If iteration $(i+1)$ is active then one of the points of S_i is π_{i+1}. However, π_{i+1} has a uniform distribution over the vertices of S_i, and in particular, if $\mathcal{E}_{i+1}$ happens then $s_{i+1} \leq s_i/2$, with probability at least half, and we will refer to such an iteration as being ***lucky***. (It is possible that $s_{i+1} < s_i$ even if $\mathcal{E}_{i+1}$ does not happen, but this is only to our benefit.) After $O(\log n)$ lucky iterations the set S_i is empty, and we are done. Clearly, if both the ith and jth iteration are active, the events that they are each lucky are independent of each other. By the Chernoff inequality, after $c \log n$ active iterations, at least $\lceil \log_2 n \rceil$ iterations were lucky with high probability, implying the claim. Here c is a sufficiently large constant. ■

Interestingly, in the above proof, all we used was the monotonicity of the sets $S_1, \ldots, S_n$, and that if $\delta(v)$ changes in an iteration then the size of the set S_i shrinks by a constant factor with good probability in this iteration. This implies that there is some flexibility in deciding whether or not to initiate Dijkstra's algorithm from each vertex of the permutation, without damaging the number of times of the values of $\delta(v)$ are updated.

Theorem 11.3.5. *Given a graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$, with n vertices and m edges, the above algorithm computes an r-net of G in $O((n \log n + m) \log n)$ expected time.*

Proof: By Lemma 11.3.4, the two δ values associated with the endpoints of an edge get updated $O(\log n)$ times, in expectation, during the algorithm's execution. As such, a single edge creates $O(\log n)$ decrease-key operations in the heap maintained by the algorithm. Each such operation takes constant time if we use Fibonacci heaps to implement the algorithm. ■

11.4. Bibliographical notes

Backwards analysis was invented/discovered by Raimund Seidel, and the **QuickSort** example is taken from Seidel [Sei93]. The number of changes of the minimum result of Section 11.1 is by now folklore.

The good ordering of Section 11.2 is probably also folklore, although a similar idea was used by Mendel and Schwob [MS09] for a different problem.

Computing a net in a sparse graph, Section 11.3.2, is from [EHS14]. While backwards analysis fails to hold in this case, it provide a good intuition for the analysis, which is slightly more complicated and indirect.

Chapter 12

Discrepancy and Derandomization

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
December 10, 2019

> "Shortly after the celebration of the four thousandth anniversary of the opening of space, Angary J. Gustible discovered Gustible's planet. The discovery turned out to be a tragic mistake.
> Gustible's planet was inhabited by highly intelligent life forms. They had moderate telepathic powers. They immediately mind-read Angary J. Gustible's entire mind and life history, and embarrassed him very deeply by making up an opera concerning his recent divorce."
>
> Gustible's Planet, Cordwainer Smith

12.1. Discrepancy

Consider a set system $(\mathsf{X}, \mathcal{R})$, where $n = |\mathsf{X}|$, and $\mathcal{R} \subseteq 2^{\mathsf{X}}$. A natural task is to partition X into two sets S, T, such that for any range $\mathbf{r} \in \mathcal{R}$, we have that $\chi(\mathbf{r}) = \big||S \cap \mathbf{r}| - |T \cap \mathbf{r}|\big|$ is minimized. In a perfect partition, we would have that $\chi(\mathbf{r}) = 0$ – the two sets S, T partition every range perfectly in half. A natural way to do so, is to consider this as a coloring problem – an element of X is colored by $+1$ if it is in S, and -1 if it is in T.

Definition 12.1.1. Consider a set system $\mathsf{S} = (\mathsf{X}, \mathcal{R})$, and let $\chi : \mathsf{X} \to \{-1, +1\}$ be a function (i.e., a coloring). The ***discrepancy*** of $\mathbf{r} \in \mathcal{R}$ is $\chi(\mathbf{r}) = |\sum_{x \in \mathbf{r}} \chi(x)|$. The ***discrepancy of*** χ is the maximum discrepancy over all the ranges – that is

$$\operatorname{disc}(\chi) = \max_{\mathbf{r} \in \mathcal{R}} \chi(\mathbf{r}).$$

The ***discrepancy*** of S is

$$\operatorname{disc}(\mathsf{S}) = \min_{\chi : \mathsf{X} \to \{-1,+1\}} \operatorname{disc}(\chi).$$

Bounding the discrepancy of a set system is quite important, as it provides a way to shrink the size of the set system, while introducing small error. Computing the discrepancy of a set system is generally quite challenging. A rather decent bound follows by using random coloring.

Definition 12.1.2. For a vector $\mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{R}^n$, $\|\mathbf{v}\|_\infty = \max_i |v_i|$.

For technical reasons, it is easy to think about the set system as an incidence matrix.

Definition 12.1.3. For a $m \times n$ a binary matrix M (i.e., each entry is either 0 or 1), consider a vector $\mathbf{b} \in \{-1, +1\}^n$. The ***discrepancy*** of $\mathbf{b}$ is $\|\mathsf{M}\mathbf{b}\|_\infty \leq 4\sqrt{n \log m}$.

Theorem 12.1.4. *Let* M *be an* $n \times n$ *binary matrix (i.e., each entry is either* 0 *or* 1*), then there always exists a vector* $\mathbf{b} \in \{-1,+1\}^n$*, such that* $\|\mathsf{M}\mathbf{b}\|_\infty \leq 4\sqrt{n \log n}$*. Specifically, a random coloring provides such a coloring with high probability.*

Proof: Let $v = (v_1, \ldots, v_n)$ be a row of M. Chose a random $\mathbf{b} = (b_1, \ldots, b_n) \in \{-1,+1\}^n$. Let $i_1, \ldots, i_\tau$ be the indices such that $v_{i_j} = 1$, and let

$$Y = \langle v, \mathbf{b} \rangle = \sum_{i=1}^{n} v_i b_i = \sum_{j=1}^{\tau} v_{i_j} b_{i_j} = \sum_{j=1}^{\tau} b_{i_j}.$$

As such Y is the sum of m independent random variables that accept values in $\{-1,+1\}$. Clearly,

$$\mathbb{E}[Y] = \mathbb{E}\big[\langle v, \mathbf{b} \rangle\big] = \mathbb{E}\Big[\sum_i v_i b_i\Big] = \sum_i \mathbb{E}[v_i b_i] = \sum_i v_i \, \mathbb{E}[b_i] = 0.$$

By Chernoff inequality and the symmetry of Y, we have that, for $\Delta = 4\sqrt{n \ln m}$, it holds

$$\mathbb{P}\big[|Y| \geq \Delta\big] = 2\,\mathbb{P}\big[\langle v, \mathbf{b} \rangle \geq \Delta\big] = 2\,\mathbb{P}\Big[\sum_{j=1}^{\tau} b_{i_j} \geq \Delta\Big] \leq 2\exp\left(-\frac{\Delta^2}{2\tau}\right) = 2\exp\left(-8\frac{n \ln m}{\tau}\right) \leq \frac{2}{m^8}.$$

Thus, the probability that any entry in $\mathsf{M}b$ exceeds $4\sqrt{n \ln}$, is smaller than $2/m^7$. Thus, with probability at least $1 - 2/m^7$, all the entries of $\mathsf{M}b$ have absolute value smaller than $4\sqrt{n \ln m}$.

In particular, there exists a vector $\mathbf{b} \in \{-1,+1\}^n$ such that $\| \mathsf{M}\mathbf{b} \|_\infty \leq 4\sqrt{n \ln m}$. ∎

We might spend more time on discrepancy later on – it is a fascinating topic, well worth its own course.

12.2. The Method of Conditional Probabilities

In previous lectures, we encountered the following problem.

Problem 12.2.1 (Set Balancing/Discrepancy). Given a binary matrix M of size $n \times n$, find a vector $\mathbf{v} \in \{-1,+1\}^n$, such that $\|\mathsf{M}\mathbf{v}\|_\infty$ is minimized.

Using random assignment and the Chernoff inequality, we showed that there exists $\mathbf{v}$, such that $\|\mathsf{M}\mathbf{v}\|_\infty \leq 4\sqrt{n \ln n}$. Can we derandomize this algorithm? Namely, can we come up with an efficient *deterministic* algorithm that has low discrepancy?

To derandomize our algorithm, construct a computation tree of depth n, where in the ith level we expose the ith coordinate of $\mathbf{v}$. This tree T has depth n. The root represents all possible random choices, while a node at depth i, represents all computations when the first i bits are fixed. For a node $v \in T$, let $P(v)$ be the probability that a random computation starting from v succeeds – here randomly assigning the remaining bits can be interpreted as a random walk down the tree to a leaf.

Formally, the algorithm is ***successful*** if ends up with a vector $\mathbf{v}$, such that $\|\mathsf{M}\mathbf{v}\|_\infty \leq 4\sqrt{n \ln n}$.

Let v_l and v_r be the two children of v. Clearly, $P(v) = (P(v_l)+P(v_r))/2$. In particular, $\max(P(v_l), P(v_r)) \geq P(v)$. Thus, if we could compute $P(\cdot)$ quickly (and deterministically), then we could derandomize the algorithm.

Let C_m^+ be the bad event that $\mathbf{r}_m \cdot \mathbf{v} > 4\sqrt{n \log n}$, where $\mathbf{r}_m$ is the mth row of M. Similarly, C_m^- is the bad event that $\mathbf{r}_m \cdot \mathbf{v} < -4\sqrt{n \log n}$, and let $C_m = C_m^+ \cup C_m^-$. Consider the probability, $\mathbb{P}\left[C_m^+ \,\middle|\, \mathbf{v}_1, \ldots, \mathbf{v}_k\right]$ (namely, the first k coordinates of $\mathbf{v}$ are specified). Let $\mathbf{r}_m = (r_1, \ldots, r_n)$. We have that

$$\mathbb{P}\left[C_m^+ \,\middle|\, \mathbf{v}_1, \ldots, \mathbf{v}_k\right] = \mathbb{P}\left[\sum_{i=k+1}^{n} \mathbf{v}_i r_i > 4\sqrt{n \log n} - \sum_{i=1}^{k} \mathbf{v}_i r_i\right] = \mathbb{P}\left[\sum_{i \geq k+1, r_i \neq 0} \mathbf{v}_i r_i > L\right] = \mathbb{P}\left[\sum_{i \geq k+1, r_i = 1} \mathbf{v}_i > L\right],$$

where $L = 4\sqrt{n \log n} - \sum_{i=1}^{k} \mathbf{v}_i r_i$ is a known quantity (since $\mathbf{v}_1, \ldots, \mathbf{v}_k$ are known). Let $V = \sum_{i \geq k+1, r_i = 1} 1$. We have,

$$\mathbb{P}\left[C_m^+ \,\middle|\, \mathbf{v}_1, \ldots, \mathbf{v}_k\right] = \mathbb{P}\left[\sum_{\substack{i \geq k+1 \\ \alpha_i = 1}} (\mathbf{v}_i + 1) > L + V\right] = \mathbb{P}\left[\sum_{\substack{i \geq k+1 \\ \alpha_i = 1}} \frac{\mathbf{v}_i + 1}{2} > \frac{L+V}{2}\right],$$

The last quantity is the probability that in V flips of a fair 0/1 coin one gets more than $(L+V)/2$ heads. Thus,

$$P_m^+ = \mathbb{P}\left[C_m^+ \,\middle|\, \mathbf{v}_1, \ldots, \mathbf{v}_k\right] = \sum_{i=\lceil (L+V)/2 \rceil}^{\mathsf{V}} \binom{\mathsf{V}}{i} \frac{1}{2^n} = \frac{1}{2^n} \sum_{i=\lceil (L+V)/2 \rceil}^{\mathsf{V}} \binom{\mathsf{V}}{i}.$$

This implies, that we can compute P_m^+ in polynomial time! Indeed, we are adding $V \leq n$ numbers, each one of them is a binomial coefficient that has polynomial size representation in n, and can be computed in polynomial time (why?). One can define in similar fashion P_m^-, and let $P_m = P_m^+ + P_m^-$. Clearly, P_m can be computed in polynomial time, by applying a similar argument to the computation of $P_m^- = \mathbb{P}\left[C_m^- \,\middle|\, \mathbf{v}_1, \ldots, \mathbf{v}_k\right]$.

For a node $v \in T$, let $\mathbf{v}_v$ denote the portion of $\mathbf{v}$ that was fixed when traversing from the root of T to v. Let $P(v) = \sum_{m=1}^{n} \mathbb{P}\left[C_m \,\middle|\, \mathbf{v}_v\right]$. By the above discussion $P(v)$ can be computed in polynomial time. Furthermore, we know, by the previous result on discrepancy that $P(r) < 1$ (that was the bound used to show that there exist a good assignment).

As before, for any $v \in T$, we have $P(v) \geq \min(P(v_l), P(v_r))$. Thus, we have a polynomial *deterministic* algorithm for computing a set balancing with discrepancy smaller than $4\sqrt{n \log n}$. Indeed, set $v = root(T)$. And start traversing down the tree. At each stage, compute $P(v_l)$ and $P(v_r)$ (in polynomial time), and set v to the child with lower value of $P(\cdot)$. Clearly, after n steps, we reach a leaf, that corresponds to a vector $\mathbf{v}'$ such that $\|A\mathbf{v}'\|_\infty \leq 4\sqrt{n \log n}$.

Theorem 12.2.2. *Using the method of conditional probabilities, one can compute in polynomial time in n, a vector $\mathbf{v} \in \{-1, 1\}^n$, such that $\|A\mathbf{v}\|_\infty \leq 4\sqrt{n \log n}$.*

Note, that this method might fail to find the best assignment.

12.3. Bibliographical Notes

There is a lot of nice work on discrepancy in geometric settings. See the books [Cha01, Mat99].

Chapter 13

Dimension Reduction

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
December 10, 2019

13.1. Introduction to dimension reduction

Given a set P of n points in $\mathbb{R}^d$, we need nd numbers to describe them. In many scenarios, d might be quite large, or even larger than n. If we care only about the distances between any pairs of points, then all we need to store are the pairwise distances between the points. This would require roughly n^2 numbers, if we just write down the distance matrix.

But can we do better? (I.e., use less space.) A natural idea is to reduce the dimension of the points. Namely, replace the ith point $\mathsf{p}_i \in P$, by a point $\mathsf{q}_i \in \mathbb{R}^k$, where $k \ll d$ and $k \ll n$. We would like k to be small. If we can do that, then we compress the data from size dn to size kn, which might be a large compression.

Of course, one can do such compression of information without losing some information. In particular, we are willing to let the distances to be a bit off. Formally, we would like to have the property that $(1-\varepsilon)\left\|\mathsf{p}_i - \mathsf{p}_j\right\| \leq \left\|\mathsf{q}_i - \mathsf{q}_j\right\| \leq (1+\varepsilon)\left\|\mathsf{p}_i - \mathsf{p}_j\right\|$, for all i, j, where q_i is the image of $\mathsf{p}_i \in P$ after the dimension reduction.

To this end, we generate a random matrix M of dimensions $d \times k$, where $k = \Theta(\varepsilon^{-2}\log n)$ (the exact details of how to generate this matrix are below, but informally every entry is going to be picked from a normal distribution and scaled appropriately). We then set $\mathsf{q}_i = \mathsf{M}\mathsf{p}_i$, for all $\mathsf{p}_i \in P$.

Before dwelling on the details, we need to better understand the normal distribution.

13.2. Normal distribution

The ***standard normal distribution*** has

$$f(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-x^2/2\right) \tag{13.1}$$

as its density function. We denote that X is distributed according to such distribution, using $X \sim \mathbf{N}(0,1)$. It is depicted in Figure 13.1.

Somewhat strangely, it would be convenient to consider two such independent variables X and Y together. Their probability space (X, Y) is the plane, and it defines a two dimensional density function

$$g(x,y) = f(x)f(y) = \frac{1}{2\pi} \exp\left(-(x^2+y^2)/2\right). \tag{13.2}$$

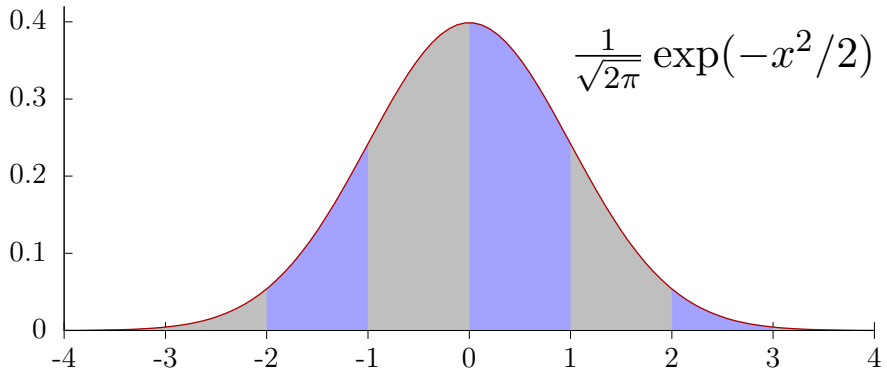


Figure 13.1

The key property of this function is that $g(x,y) = g(x',y') \iff \|(x,y)\|^2 = x^2+y^2 = \|(x',y')\|^2$. Namely, $g(x,y)$ is symmetric around the origin (i.e., all the points in the same distance from the origin have the same density). We next use this property in verifying that $f(\cdot)$ it is indeed a valid density function.

Lemma 13.2.1. *We have* $I = \int_{-\infty}^{\infty} f(x)\,\mathrm{d}x = 1$.

Proof: Observe that

$$I^2 = \left(\int_{x=-\infty}^{\infty} f(x)\ \mathrm{d}x\right)^2 = \left(\int_{x=-\infty}^{\infty} f(x)\ \mathrm{d}x\right)\left(\int_{y=-\infty}^{\infty} f(y)\ \mathrm{d}y\right) = \int_{x=-\infty}^{\infty}\int_{y=-\infty}^{\infty} f(x)f(y)\,dx\,dy$$
$$= \int_{x=-\infty}^{\infty}\int_{y=-\infty}^{\infty} g(x,y)\,\mathrm{d}x\,\mathrm{d}y.$$

Change the variables to $x = r\cos\alpha$, $y = r\sin\alpha$, and observe that the determinant of the Jacobian is

$$J = \det\begin{vmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial \alpha} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial \alpha} \end{vmatrix} = \det\begin{vmatrix} \cos\alpha & -r\sin\alpha \\ \sin\alpha & r\cos\alpha \end{vmatrix} = r\left(\cos^2\alpha + \sin^2\alpha\right) = r.$$

As such,

$$I^2 = \frac{1}{2\pi}\int_{r=0}^{\infty}\int_{\alpha=0}^{2\pi} \exp\left(-\frac{r^2}{2}\right)|J|\ d\alpha\,dr = \frac{1}{2\pi}\int_{r=0}^{\infty}\int_{\alpha=0}^{2\pi} \exp\left(-\frac{r^2}{2}\right) r\,d\alpha\,dr$$
$$= \int_{r=0}^{\infty} \exp\left(-\frac{r^2}{2}\right) r\,dr = \left[-\exp\left(-\frac{r^2}{2}\right)\right]_{r=0}^{r=\infty} = -\exp(-\infty) - (-\exp(0)) = 1. \qquad \blacksquare$$

Lemma 13.2.2. *For* $X \sim \mathbf{N}(0,1)$, *we have that* $\mathbb{E}[X] = 0$ *and* $\mathbb{V}[X] = 1$.

Proof: The density function of X, see Eq. (13.2) is symmetric around 0, which implies that $\mathbb{E}[X] = 0$. As for the variance, we have $\mathbb{V}[X] = \mathbb{E}\left[X^2\right] - (\mathbb{E}[X])^2 = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} x^2\exp(-x^2/2)\,\mathrm{d}X$. Observing that

$$x^2\exp\left(-x^2/2\right) = \left(-x\exp(-x^2/2)\right)' + \exp\left(-x^2/2\right),$$

implies (using integration by guessing) that

$$\mathbb{V}[X] = \frac{1}{\sqrt{2\pi}}\left[-x\exp(-x^2/2)\right]_{-\infty}^{\infty} + \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}\exp(-x^2/2)\,\mathrm{d}x = 0 + 1 = 1. \qquad \blacksquare$$

13.2.1. The standard multi-dimensional normal distribution

The ***multi-dimensional normal distribution***, denoted by $\mathbf{N}^d$, is the distribution in $\mathbb{R}^d$ that assigns a point $\mathsf{p} = (\mathsf{p}_1, \ldots, \mathsf{p}_d)$ the density

$$g(\mathsf{p}) = \frac{1}{(2\pi)^{d/2}} \exp\Bigl(-\frac{1}{2}\sum_{i=1}^{d} \mathsf{p}_i^2\Bigr).$$

It is easy to verify, using the above, that $\int_{\mathbb{R}^d} g(\mathsf{p})d\mathsf{p} = 1$. Furthermore, we have the following useful but easy properties.[①]

Lemma 13.2.3. *(A) Consider d independent variables $X_1, \ldots, X_d \sim \mathbf{N}(0,1)$, the point $\mathsf{q} = (X_1, \ldots, X_d)$ has the multi-dimensional normal distribution $\mathbf{N}^d$.*

(B) The multi-dimensional normal distribution is symmetric; that is, for any two points $\mathsf{p}, \mathsf{q} \in \mathbb{R}^d$ such that $\|\mathsf{p}\| = \|\mathsf{q}\|$ we have that $g(\mathsf{p}) = g(\mathsf{q})$, where $g(\cdot)$ is the density function of the multi-dimensional normal distribution $\mathbf{N}^d$.

(C) The projection of the normal distribution on any direction (i.e., any vector of length 1) is a one-dimensional normal distribution.

Proof: (A) Let $f(\cdot)$ denote the density function of $\mathbf{N}(0,1)$, and observe that the density function of q is $f(X_1)f(X_2)\cdots f(X_d), = \frac{1}{\sqrt{2\pi}}\exp\bigl(-X_1^2/2\bigr)\cdots\frac{1}{\sqrt{2\pi}}\exp\bigl(-X_d^2/2\bigr)$, which readily implies the claim.

(B) Readily follows from observing that $g(\mathsf{p}) = \frac{1}{(2\pi)^{d/2}}\exp\bigl(-\|\mathsf{p}\|^2/2\bigr)$.

(C) Let $\mathsf{p} = (X_1, \ldots, X_d)$, where $X_1, \ldots, X_d \sim \mathbf{N}(0,1)$. Let v be any unit vector in $\mathbb{R}^d$, and observe that by the symmetry of the density function, we can (rigidly) rotate the space around the origin in any way we want, and the measure of sets does not change. In particular rotate space so that v becomes the unit vector $(1, 0, \ldots, 0)$. We have that

$$\mathbb{P}[\langle v, \mathsf{p}\rangle \le \alpha] = \mathbb{P}[\langle (1,0,\ldots,0), \mathsf{p}\rangle \le \alpha] = \mathbb{P}[X_1 \le \alpha],$$

which implies that $\langle v, \mathsf{p}\rangle \sim X_1 \sim \mathbf{N}(0,1)$. ∎

The generalized multi-dimensional distribution is a ***Gaussian***. Fortunately, we only need the simpler notion.

13.3. Dimension reduction

13.3.1. The construction

The input is a set $P \subseteq \mathbb{R}^d$ of n points (where d is potentially very large), and let $\varepsilon > 0$ be an approximation parameter. For

$$k = \bigl\lceil 24\varepsilon^{-2}\ln n \bigr\rceil \tag{13.3}$$

we pick k vectors $u_1, \ldots, u_k$ independently from the d-dimensional normal distribution $\mathbf{N}^d$. Given a point $\mathsf{p} \in \mathbb{R}^d$, its image is

$$h(v) = \frac{1}{\sqrt{k}}\Bigl(\langle u_1, \mathsf{p}\rangle, \cdots, \langle u_k, \mathsf{p}\rangle\Bigr).$$

[①] The normal distribution has such useful properties that it seems that the only thing normal about it is its name.

In matrix notation, let

$$\mathsf{M} = \frac{1}{\sqrt{k}} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_k \end{pmatrix}.$$

For every point $\mathsf{p}_i \in P$, we set $\mathsf{q}_i = h(\mathsf{p}_i) = \mathsf{M}\mathsf{p}_i$.

13.3.2. Analysis

13.3.2.1. A single unit vector is preserved

Consider a vector v of length one in $\mathbb{R}^d$. The natural question is what is the value of k needed, so that the length of $h(v)$ is a good approximation to v. Since $\langle u_i, v\rangle \sim \mathbf{N}(0,1)$, by Lemma 13.2.3, this question can boil down to the following: Given k variables $X_1, \ldots, X_k \sim \mathbf{N}(0,1)$, sampled independently, how concentrated is the random variable

$$Y = \|(X_1, \ldots, X_k)\|^2 = \sum_{i=1}^{k} X_i^2.$$

We have that $\mathbb{E}[Y] = k\,\mathbb{E}\left[X_i^2\right] = k\,\mathbb{V}[X_i] = k$, since $X_i \sim \mathbf{N}(0,1)$, for any i. The distribution of Y is known as the ***chi-square distribution with k degrees of freedom***.

Lemma 13.3.1. *Let $\varphi \in (0,1)$, and $\varepsilon \in (0, 1/2)$ be parameters, and let $k \geq \left\lceil \frac{16}{\varepsilon^2} \ln \frac{2}{\varphi} \right\rceil$ be an integer. Then, for k independent random variables $X_1, \ldots, X_k \sim \mathbf{N}(0,1)$, we have that $Z = \sum_i X_i^2/k$ is strongly concentrated. Formally, we have that $\mathbb{P}[Z \leq 1+\varepsilon] \geq 1 - \varphi$.*

Proof: Arguing as in the proof of Chernoff's inequality, using $t = \varepsilon/4 < 1/2$, we have

$$\mathbb{P}[Z \geq 1+\varepsilon] \leq \mathbb{P}\left[\exp(tkZ) \geq \exp\left(tk(1+\varepsilon)\right)\right] \leq \frac{\mathbb{E}[\exp(tkZ)]}{\exp\left(tk(1+\varepsilon)\right)} = \prod_{i=1}^{k} \frac{\mathbb{E}\left[\exp(tX_i^2)\right]}{\exp\left(t(1+\varepsilon)\right)}$$

Using substitution (i.e., $y = \frac{x}{\sqrt{1-2t}}$), and $t = \varepsilon/4$, we have

$$\begin{aligned}
\mathbb{E}\left[\exp(tX_i^2)\right] &= \int_{-\infty}^{\infty} \frac{\exp(tx^2)}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \mathrm{d}x = \frac{1}{\sqrt{2\pi}} \int \exp\left(-(1-2t)\frac{x^2}{2}\right) \mathrm{d}x \\
&= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \frac{1}{\sqrt{1-2t}} \exp\left(-\frac{y^2}{2}\right) \mathrm{d}y = \frac{1}{\sqrt{1-2t}} = \frac{1}{\sqrt{1-\varepsilon/2}} \leq \exp\left(\frac{1}{2}\sum_{i=1}^{\infty}\left(\frac{\varepsilon}{2}\right)^i\right).
\end{aligned}$$

since $\frac{1}{1-z} = \sum_{i=0}^{\infty} z^i$, for $0 \leq z < 1$, and thus

$$\frac{1}{1-\varepsilon/2} = \sum_i \left(\frac{\varepsilon}{2}\right)^i \leq \left(1 + \frac{1}{2}\sum_{i=1}^{\infty}\left(\frac{\varepsilon}{2}\right)^i\right)^2 \leq \exp\left(\frac{1}{2}\sum_{i=1}^{\infty}\left(\frac{\varepsilon}{2}\right)^i\right)^2.$$

As such, we have

$$\mathbb{P}[Z \geq 1+\varepsilon] \leq \exp\left(\frac{1}{2}\sum_{i=1}^{\infty}\left(\frac{\varepsilon}{2}\right)^i - \frac{\varepsilon}{4}(1+\varepsilon)\right)^k = \exp\left(-\frac{\varepsilon^2}{8} + \frac{1}{2}\sum_{i=3}^{\infty}\left(\frac{\varepsilon}{2}\right)^i\right)^k \leq \exp\left(-\frac{k\varepsilon^2}{16}\right) \leq \frac{\varphi}{2},$$

since, for $\varepsilon < 1/2$, we have $\frac{1}{2}\sum_{i=3}^{\infty}\left(\frac{\varepsilon}{2}\right)^i \leq \left(\frac{\varepsilon}{2}\right)^3 \leq \frac{\varepsilon^2}{16}$. The last step in the above inequality follows by substituting in the lower bound on the value of k. ∎

The other direction we need follows in a similar fashion. We state the needed result without proof [LM00, Lemma 1] (which also yields better constants):

Lemma 13.3.2. *Let $Y_1, \ldots, Y_k$ be k independent random variables with $Y_i \sim \mathbf{N}(0,1)$. Let $Z = \sum_{i=1}^{k} Y_i^2/k$. For any $x > 0$, we have that*

$$\mathbb{P}\left[Z \leq 1 - 2\sqrt{x/k}\right] \leq \exp(-x) \qquad \text{and} \qquad \mathbb{P}\left[Z \geq 1 + 2\sqrt{x/k} + 2x/k\right] \leq \exp(-x).$$

For our purposes, we require that $\exp(-x) \leq \varphi/2$, which implies $x = \ln(2/\varphi)$. We further require that $2\sqrt{x/k} \leq \varepsilon$ and $2\sqrt{x/k} + 2x/k \leq \varepsilon$, which hold for $k = 8\varepsilon^{-2}\ln\frac{2}{\varphi}$, for $\varepsilon \leq 1$. We thus get the following result.

Corollary 13.3.3. *Let $\varphi \in (0,1)$, and $\varepsilon \in (0, 1/2)$ be parameters, and let $k \geq \left\lceil \frac{8}{\varepsilon^2}\ln\frac{2}{\varphi}\right\rceil$ be an integer. Then, for k independent random variables $X_1, \ldots, X_k \sim \mathbf{N}(0,1)$, we have for $Z = \sum_i X_i^2/k$ that that $\mathbb{P}[1-\varepsilon \leq Z \leq 1+\varepsilon] \geq 1 - \varphi$.*

Remark 13.3.4. The result of Corollary 13.3.3 is surprising. It says that if we pick a point according to the k-dimensional normal distribution, then its distance to the origin is strongly concentrated around $\sqrt{k}$. Namely, the normal distribution "converges" to a sphere, as the dimension increases. The mind boggles.

Lemma 13.3.5. *Let v be a unit vector in $\mathbb{R}^d$, then*

$$\mathbb{P}[1-\varepsilon \leq \|\mathsf{M}v\| \leq 1+\varepsilon] \geq 1 - \frac{1}{n^2}.$$

Proof: Observe that if for a number x, if $1-\varepsilon \leq x^2 \leq 1+\varepsilon$, then $1-\varepsilon \leq x \leq 1+\varepsilon$. As such, the claim holds if $1-\varepsilon \leq \|\mathsf{M}v\|^2 \leq 1+\varepsilon$. By Corollary 13.3.3, setting $\varphi = 1/n^2$, we need

$$k \geq 8\varepsilon^{-2}\ln(2/\varphi) = 8\varepsilon^{-2}\ln(2n^2) = 24\varepsilon^{-2}\ln n,$$

which holds for the value picked for k in Eq. (13.3). ∎

13.3.3. All pairwise distances are preserved

Lemma 13.3.6. *With probability at least half, for all points $\mathsf{p}, \mathsf{p}' \in P$, we have that*

$$(1-\varepsilon)\|\mathsf{p}-\mathsf{p}'\| \leq \|\mathsf{M}\mathsf{p} - \mathsf{M}\mathsf{q}\| \leq (1+\varepsilon)\|\mathsf{p}-\mathsf{q}\|.$$

Proof: The key observation is that M is a linear operator. As such, let $v = (\mathsf{p}-\mathsf{p}')/\|\mathsf{p}-\mathsf{p}'\|$ be a unit vector, and observe that

$$(1-\varepsilon)\|\mathsf{p}-\mathsf{p}'\| \leq \|\mathsf{M}\mathsf{p} - \mathsf{M}\mathsf{q}\| \leq (1+\varepsilon)\|\mathsf{p}-\mathsf{q}\| \qquad \Longleftrightarrow \qquad (1-\varepsilon)\|v\| \leq \|\mathsf{M}v\| \leq (1+\varepsilon)\|v\|.$$

The probability the later condition does not hold is at most $1/n^2$, by Lemma 13.3.5. As such, for all possible pairs of points, the probability of failure is $\binom{n}{2}\cdot\frac{1}{n^2} \leq 1/2$, as claimed. ∎

We thus got the famous JL-Lemma.

Theorem 13.3.7 (The Johnson-Lindenstrauss Lemma). *Given a set P of n points in $\mathbb{R}^d$, and a parameter ε, one can reduce the dimension of P to $k = O(\varepsilon^{-2}\log n)$ dimensions, such that all pairwise distances are $1 \pm \varepsilon$ preserved.*

13.4. Even more on the normal distribution

The following is not used anywhere in the above, and is provided as additional information about the normal distribution.

Lemma 13.4.1. *Let $X \sim \mathbf{N}(0,1)$, and let $\sigma > 0$ and μ be two real numbers. The random variable $Y = \sigma X + \mu$ has the density function*

$$f_{\mu,\sigma}(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right). \tag{13.4}$$

The variable Y has the **normal distribution** *with variance σ^2, and expectation μ, denoted by $Y \sim \mathbf{N}(\mu, \sigma^2)$.*

Proof: We have $\mathbb{P}[Y \le \alpha] = \mathbb{P}[\sigma X + \mu \le \alpha] = \mathbb{P}\left[X \le \frac{\alpha-\mu}{\sigma}\right] = \int_{y=-\infty}^{(\alpha-\mu)/\sigma} f(y)\,\mathrm{d}y$, where $f(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-x^2/2\right)$. Substituting $y = (x-\mu)/\sigma$, and observing that $\mathrm{d}y/\mathrm{d}x = 1/\sigma$, we have

$$\mathbb{P}[Y \le \alpha] = \int_{x=-\infty}^{\alpha} f\left(\frac{x-\mu}{\sigma}\right)\frac{1}{\sigma}\,\mathrm{d}x = \frac{1}{\sqrt{2\pi}\sigma}\int_{x=-\infty}^{\alpha} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)\mathrm{d}x,$$

as claimed.

As for the second part, observe that $\mathbb{E}[Y] = \mathbb{E}[\sigma X + \mu] = \sigma\,\mathbb{E}[X] + \mu = \mu$ and $\mathbb{V}[Y] = \mathbb{V}[\sigma X + \mu] = \mathbb{V}[\sigma X] = \sigma^2\,\mathbb{V}[X] = \sigma^2$. ∎

Lemma 13.4.2. *Consider two independent variables $X \sim \mathbf{N}(0,1)$ and $Y \sim \mathbf{N}(0,1)$. For $\alpha, \beta > 0$, we have $Z = \alpha X + \beta Y \sim \mathbf{N}(0, \sigma^2)$, where $\sigma = \sqrt{\alpha^2 + \beta^2}$.*

Proof: Consider the region in the plane $H^- = \left\{(x,y) \in \mathbb{R}^2 \;\middle|\; \alpha x + \beta y \le z\right\}$ – this is a halfspace bounded by the line $\ell \equiv \alpha x + \beta y = z$. This line is orthogonal to the vector $(-\beta, \alpha)$. We have that $\ell \equiv \frac{\alpha}{\sigma}x + \frac{\beta}{\sigma}y = \frac{z}{\sigma}$. Observe that $\left\|\left(\frac{\alpha}{\sigma}, \frac{\beta}{\sigma}\right)\right\| = 1$, which implies that the distance of ℓ from the origin is $d = z/\sigma$.

Now, we have

$$\mathbb{P}[Z \le z] = \mathbb{P}[\alpha X + \beta Y \le z] = \mathbb{P}[H^-] = \int_{p=(x,y)\in H^-} g(x,y)\,\mathrm{d}p,$$

see Eq. (13.2). Since, the two dimensional density function g is symmetric around the origin. any halfspace containing the origin, which its boundary is in distance d from the origin, has the same probability. In particular, consider the halfspace $T = \left\{(x,y) \in \mathbb{R}^2 \;\middle|\; x \le d\right\}$. We have that

$$\begin{aligned}\mathbb{P}[Z \le z] = \mathbb{P}[H^-] = \mathbb{P}[T] = \mathbb{P}[X \le d] &= \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{d} \exp\left(-\frac{x^2}{2}\right)\mathrm{d}x = \frac{1}{\sqrt{2\pi}}\int_{y=-\infty}^{z} \exp\left(-\frac{y^2}{2\sigma^2}\right)\frac{\mathrm{d}x}{\mathrm{d}y}\,\mathrm{d}y,\\ &= \frac{1}{\sqrt{2\pi}\sigma}\int_{y=-\infty}^{z} \exp\left(-\frac{y^2}{2\sigma^2}\right)\mathrm{d}y,\end{aligned}$$

by change of variables $x = y/\sigma$, and observing that $\mathrm{d}x/\mathrm{d}y = 1/\sigma$. By Eq. (13.4), the above integral is the probability of a variable distributed $\mathbf{N}(0, \sigma^2)$ to be smaller than z, establishing the claim. ∎

Lemma 13.4.3. *Consider two independent variables $X \sim \mathbf{N}(\mu_1, \sigma_1^2)$ and $Y \sim \mathbf{N}(\mu_2, \sigma_2^2)$. We have $Z = X + Y \sim \mathbf{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$,*

Proof: Let $\widehat{X} \sim \mathbf{N}(0,1)$ and $\widehat{Y} \sim \mathbf{N}(0,1)$, and observe that we can write $X = \sigma_1 \widehat{X} + \mu_1$ and $Y = \sigma_2 \widehat{Y} + \mu_2$. As such, we have

$$Z = X + Y = \sigma_1 \widehat{X} + \sigma_2 \widehat{Y} + \mu_1 + \mu_2.$$

The variable $W = \sigma_1 \widehat{X} + \sigma_2 \widehat{Y} \sim \mathbf{N}\big(0, \sigma_1^2 + \sigma_2^2\big)$, by Lemma 13.4.2. Adding $\mu_1 + \mu_2$ to W, just shifts its expectation, implying the claim. ■

13.5. Some calculations

Let $X \sim \mathbf{N}(0,1)$. Then

$$\mathbb{P}\big[X^2 \leq \alpha\big] = \int_{x=-\sqrt{\alpha}}^{\sqrt{\alpha}} f(x)\,\mathrm{d}x = \int_{x=0}^{\sqrt{\alpha}} 2f(x)\,\mathrm{d}x = \int_{y=0}^{\alpha} \frac{1}{2\sqrt{y}} 2f(\sqrt{y})\,\mathrm{d}y = \int_{y=0}^{\alpha} \frac{1}{\sqrt{2\pi y}} \exp(-y/2)\,\mathrm{d}y$$

13.6. Bibliographical notes

The original result is due to Johnson and Lindenstrauss [JL84]. By now there are many proofs of this lemma. Our proof follows class notes of Anupam Gupta, which in turn follows Indyk and Motwani [IM98],

Chapter 14

Streaming and the Multipass Model

I don't know why it should be, I am sure; but the sight of another man asleep in bed when I am up, maddens me. It seems to me so shocking to see the precious hours of a man's life - the priceless moments that will never come back to him again - being wasted in mere brutish sleep.

Jerome K. Jerome, Three men in a boat

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
December 10, 2019

14.1. Reservoir sampling: Fishing a sample from a stream

Imagine that you are given a stream of elements $s_1, s_2, \ldots$, and you need to sample k numbers from this stream (say, without repetition) – assume that you do not know the length of the stream in advance, and furthermore, you have only $O(k)$ space available for you. How to do that efficiently?

There are two natural schemes:

(A) Whenever an element arrives, generate a random number for it in the range $[0, 1]$. Maintain a heap with the k elements with the lowest priority. Implemented naively this requires $O(\log k)$ comparisons after each insertion, but it is not difficult to improve this to $O(1)$ comparisons in the amortized sense per insertion. Clearly, the resulting set is the desired random sample

(B) Let S_t be the random sample maintained in the tth iteration. When the ith element arrives, the algorithm flip a coin that is heads with probability $\min(1, k/i)$. If the coin is heads then it inserts s_i to S_{i-1} to get S_i. If S_{i-1} already have k elements, then first randomly delete one of the elements.

Theorem 14.1.1. *Given a stream of elements, one can uniformly sample k elements (without repetition), from the stream using $O(k)$ space, where $O(1)$ time is spent for handling each incoming element.*

Proof: We implement the scheme (B) above. We only need to argue that this is a uniform random sample. The claim trivially hold for $i = k$. So assume the claim holds for $i < t$, and we need to prove that the set after getting tth element is still a uniform random sample.

So, consider a specific set $K \subseteq \{s_1, \ldots, s_t\}$ of k elements. The probability of K to be a random sample of size k from a set of t elements is $1/\binom{t}{k}$. We need to argue that this probability remains the same for this scheme.

So, if $s_t \notin K$, then we have

$$\mathbb{P}[K = S_t] = \mathbb{P}[K = S_{t-1} \text{ and } s_t \text{ was not inserted}] = \frac{1}{\binom{t-1}{k}}\left(1 - \frac{k}{t}\right) = \frac{k!(t-1-k)!(t-k)}{(t-1)!t} = \frac{1}{\binom{t}{k}}.$$

If $s_t \in K$, then

$$\mathbb{P}\big[K = S_t\big] = \mathbb{P}\left[\begin{array}{c} K \setminus \{s_t\} \subseteq S_{t-1}, \\ s_t \text{ was inserted} \\ \text{and } S_{t-1} \setminus K \text{ thrown out of } S_{t-1} \end{array}\right] = \frac{t-1-(k-1)}{\binom{t-1}{k}}\frac{k}{t}\frac{1}{k} = \frac{(t-k)k!(t-1-k)!}{(t-1)!t} = \frac{1}{\binom{t}{k}},$$

as desired. Indeed, there are $t-1-(k-1)$ subsets of size k of $\{s_1, \ldots, s_{t-1}\}$ that contains $K \setminus \{s_t\}$ – since we fix $k-1$ of the $t-1$ elements. ■

14.2. Sampling and median selection revisited

Let $B[1, \ldots, n]$ be a set of n numbers. We would like to estimate the median, without computing it outright. A natural idea, would be to pick k elements $e_1, \ldots, e_k$ randomly from B, and return their median as the guess for the median of B.

In the following, let $\operatorname{rank}(B, t)$ be the tth smallest number in the array B.

Observation 14.2.1. *For $\varepsilon \in (0,1)$, we have that $\frac{1}{1-\varepsilon} \geq 1 + \varepsilon$.*

Lemma 14.2.2. *Let $\varepsilon \in (0, 1/2)$, and let $k = \lceil \frac{12}{\varepsilon^2} \ln \frac{2}{\delta} \rceil$. Let Z be the median of the random sample of B of size k. We have that*

$$\mathbb{P}\left[\operatorname{rank}\left(B, \frac{1-\varepsilon}{2}n\right) \leq Z \leq \operatorname{rank}\left(B, \frac{1+\varepsilon}{2}n\right)\right] \geq 1 - \delta.$$

Namely, with probability at least $1-\delta$, the returned value Z is $(\varepsilon/2)n$ positions away from the true median.

Proof: Let $L = \operatorname{rank}(B, (1-\varepsilon)n/2)$. Let $X_i = 1$ if and only if $e_i \leq L$. We have that

$$\mathbb{P}[X_i = 1] = \frac{(1-\varepsilon)n/2}{n} = \frac{1-\varepsilon}{2}.$$

As such, setting $Y = \sum_{i=1}^{k} X_i$, we have

$$\mu = \mathbb{E}[Y] = \frac{1-\varepsilon}{2}k \geq \frac{k}{4} \geq \frac{3}{\varepsilon^2}\ln\frac{2}{\delta}.$$

One case is that the algorithm fails, if $Y \geq k/2$. Since $\frac{1}{1-\varepsilon} \geq 1+\varepsilon$, we have that

$$\mathbb{P}[Y \geq k/2] = \mathbb{P}\left[Y \geq \frac{1/2}{(1-\varepsilon)/2} \cdot \frac{1-\varepsilon}{2}k\right] \leq \mathbb{P}[Y \geq (1+\varepsilon)\mu] \leq \exp\left(-\frac{\varepsilon^2\mu}{3}\right) \leq \exp\left(-\frac{\varepsilon^2}{3} \cdot \frac{3}{\varepsilon^2}\ln\frac{2}{\delta}\right) \leq \frac{\delta}{2}.$$

by Chernoff's inequality (see Lemma 8.2.4).

This implies that $\mathbb{P}\big[\operatorname{rank}(B, (1-\varepsilon)n/2) > Z\big] \leq \delta/2$.

The claim now follows by realizing that by symmetry (i.e., revering the order), we have that $\mathbb{P}[Z > \operatorname{rank}(B, (1+\varepsilon)n/2)] \leq \delta/2$, and putting these two inequalities together. ■

The above already implies that we can get a good estimate for the median. We need something somewhat stronger – we state it without proof since it follows by similarly mucking around with Chernoff's inequality.

Lemma 14.2.3. *Let $\varepsilon \in (0,1/2)$, let B an array of n elements, and let $S = \{e_1, \ldots, e_k\}$ be a set of k samples picked uniformly and randomly from B. Then, for some absolute constant c, and an integer k, such that $k \geq \left\lceil \frac{c}{\varepsilon^2} \ln \frac{1}{\delta} \right\rceil$, we have that*

$$\mathbb{P}\Big[\operatorname{rank}(S, k_-) \leq \operatorname{rank}(B, n/2) \leq \operatorname{rank}\big(S, k^+\big)\Big] \geq 1 - \delta.$$

for $k_- = \lfloor (1-\varepsilon)k/2 \rfloor$, and $k^+ = \lfloor (1+\varepsilon)k/2 \rfloor$.

One can prove even a stronger statement:

$$\mathbb{P}[\operatorname{rank}(B,(1-2\varepsilon)n/2) \leq \operatorname{rank}(S,(1-\varepsilon)k/2) \leq \operatorname{rank}(B,n/2) \leq \operatorname{rank}(S,(1+\varepsilon)k/2) \leq \operatorname{rank}(B,(1+2\varepsilon)n/2)] \geq 1-$$

(the constant c would have to be slightly bigger).

14.2.1. A median selection with few comparisons

The above suggests a natural algorithm for computing the median (i.e., the element of rank $n/2$ in B). Pick a random sample S of $k = O(n^{2/3} \log n)$ elements. Next, sort S, and pick the elements L and R of ranks $(1-\varepsilon)k$ and $(1+\varepsilon)k$ in S, respectively. Next, scan the elements, and compare them to L and R, and keep only the elements that are between. In the end of this process, we have computed:

(A) α: The rank of the number L in the set B.

(B) $T = \{x \in B \mid L \leq x \leq H\}$.

Compute, by brute force (i.e., sorting) the element of rank $n/2 - \alpha$ in T. Return it as the desired median. If $n/2 - \alpha$ is negative, then the algorithm failed, and it tries again.

Lemma 14.2.4. *The above algorithm performs $2n + O(n^{2/3} \log n)$ comparisons, and reports the median. This holds with high probability.*

Proof: Set $\varepsilon = 1/n^{1/3}$, and $\delta = 1/n^{O(1)}$, and observe that Lemma 14.2.3 implies that with probability $\geq 1 - 1/\delta$, we have that the desired median is between L and H. In addition, Lemma 14.2.3 also implies that $|T| \leq 4\varepsilon n \leq 4n^{2/3}$, which readily implies the correctness of the algorithm.

As for the bound on the number of comparisons, we have, with high probability, that the number of comparisons is

$$O(|S| \log |S| + |T| \log |T|) + 2n = O\left(\sqrt{n} \log^2 n + n^{2/3} \log n\right) + 2n,$$

since deciding if an element is between L and H requires two comparisons. ∎

Lemma 14.2.5. *The above algorithm can be modified to perform $(3/2)n + O(n^{2/3} \log n)$ comparisons, and reports the median correctly. This holds with high probability.*

Proof: The trick is to randomly compare each element either first to L or first to H with equal probability. For elements that are either smaller than L or bigger than H, this requires $(3/2)n$ comparisons in expectation. Thus improving the bound from $2n$ to $(3/2)n$. ∎

Lemma 14.2.6. *Consider a stream B of n numbers, and assume we can make two passes over the data. Then, one can compute exactly the median of B using:*

(I) $O(n^{2/3})$ *space.*
(II) $1.5n + O(n^{2/3} \log n)$ *comparisons.*
The algorithm reports the median correctly, and it succeeds with high probability.

Proof: Implement the above algorithm, using the random sampling from Theorem 14.1.1. ■

Remark 14.2.7. Interestingly, one can do better if one is more careful. The basic idea is to do thinning – given two sorted sequence of sizes s, consider merging the sets, and then picking all the even rank elements into a new sequence. Clearly, the element of rank i in the output sequence, has rank $2i$ in the union of the two original sequences. A sequence that is the result of i such rounds of thinning is of rank i. We maintain $O(\log n)$ such sequences as we read the stream. At any time, we have two buffers of size s, that we fill up from the stream. Whenever the two buffers fill up, we perform the thinning operation on them, creating a sequence of rank 1.

If during this process we store two sequence of the same rank, we merge them and perform thinning on them. As such, we maintain $O(\log n)$ buffers sequences each of size s. Assume that our stream has size n, and n is a power for 2. Then in the end of process, we would have only a single sequence of rank $h = \log_2(n/s)$. By induction, it is easy to prove that an element of rank r in this sequence, has rank between $2^h(r-1)$ and $2^h r$ in the original stream.

Thus, setting $s = \sqrt{n}$, we get that after a single pass, using $O(\sqrt{n} \log n)$ space, we have a sorted sequence, where the rank of the elements is roughly $\sqrt{n}$ approximation to the true rank. We pick the two consecutive elements (or more carefully, the predecessor, and successor), and filter the stream again, keeping only the elements in between these two elements. It is to show that $O(\sqrt{n})$ would be kept, and we can extract the median using $O(\sqrt{n} \log n)$ time.

We thus got that one can compute the median in two passes using $O(\sqrt{n} \log n)$ space. It is not hard to extend this algorithm to α-passes, where the space required becomes $O(n^{1/\alpha} \log n)$.

This elegant algorithm goes back to 1980, and it is by Munro and Paterson [MP80].

14.3. Big data and the streaming model

Here, we are interested in doing some computational tasks when the amount of data we have to handle is quite large (think terabytes or larger). The main challenge in many of these cases is that even reading the data once is expensive. Running times of $O(n \log n)$ might not be acceptable. Furthermore, in many cases, we can *not* load all the data into memory.

In the ***streaming*** model, one reads the data as it comes in, but one can not afford to keep all the data. A natural example would be a internet router, which has gazillion of packets going through it every minute. We might still be interested in natural questions about these packets, but we want to do this without storing all the packets.

14.4. Heavy hitters

Imagine a stream $s_1, \ldots$, where elements might repeat, and we would like to maintain a list of elements that appear at least εn times. We present a simple but clever scheme that maintains such a list.

The algorithm. To this end, let

$$k = \lceil 1/\varepsilon \rceil .$$

At each point in time, we maintain a set S of k elements, with a counter for each element. Let S_t be the version of S after t were inserted. When s_{t+1} arrives, we increase its counter if it is already in S_t. If $|S_t| < k$, then we just insert s_{t+1} to the set, and set its counter to 1. Otherwise, $|S_t| = k$ and $s_{t+1} \notin S_t$. We then decrease all the k counters of elements in S_t by 1. If a counter of an element in S_{t+1} is zero, then we delete it from the set.

Chapter 15

Independent set – Turán's theorem

I don't know why it should be, I am sure; but the sight of another man asleep in bed when I am up, maddens me. It seems to me so shocking to see the precious hours of a man's life - the priceless moments that will never come back to him again - being wasted in mere brutish sleep.

Jerome K. Jerome, Three men in a boat

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
December 10, 2019

15.0.1. Statement & proof

I think the following proof is due to Alon and Spencer.

Theorem 15.0.1 (Turán's theorem). *Let $G = (V, E)$ be a graph. The graph G has an independent set of size $\frac{n}{1+d_G}$, where $n = |V|$ and d_G is the average vertex degree in G.*

Proof: Let $\pi = (\pi_1, \ldots, \pi_n)$ be a random permutation of the vertices of G. Pick the vertex π_i into the independent set if none of its neighbors appear before it in π. Clearly, v appears in the independent set if and only if it appears in the permutation before all its $d(v)$ neighbors. The probability for this is $1/(1 + d(v))$. Thus, the expected size of the independent set is (exactly)

$$\tau = \sum_{v \in V} \frac{1}{1 + d(v)}, \tag{15.1}$$

by linearity of expectations. Thus, by the probabilistic method, there exists an independent set in G of size at least τ.

We remain with the task of proving that $\tau \geq \frac{n}{1+d_G}$. Observe that if $x + y = \alpha$, then

$$\frac{1}{1+x} + \frac{1}{1+y} = \frac{1+x+1+y}{1+x+y+xy} = \frac{2+\alpha}{1+\alpha+xy} \geq \frac{2+\alpha}{1+\alpha+\alpha^2/4} = \frac{2(1+\alpha/2)}{(1+\alpha/2)^2} = \frac{2}{1+\alpha/2},$$

since the quantity xy is maximized when $x = y$ under the condition $x + y = \alpha$. This implies that the minimum of Eq. (15.1) is achieved if we replace $d(v)$ by the average degree in G, which implies the theorem. ∎

Following a post of this write-up on my blog, readers suggested two modifications. We present an alternative proof incorporating both suggestion.

Alternative proof of Theorem 15.0.1: We associate a charge of size $1/(d(v)+1)$ with each vertex of G. Let $\gamma(G)$ denote the total charge of the vertices of G. We prove, using induction, that there is always an independent set in G of size at least $\gamma(G)$. If G is the empty graph, then the claim trivially holds. Otherwise, assume that it holds if the graph has at most $n-1$ vertices, and consider the vertex v of lowest degree in G. The total charge of v and its neighbors is

$$\frac{1}{d(v)+1} + \sum_{uv \in E} \frac{1}{d(u)+1} \leq \frac{1}{d(v)+1} + \sum_{uv \in E} \frac{1}{d(v)+1} = \frac{d(v)+1}{d(v)+1} = 1,$$

since $d(u) \geq d(v)$, for all $uv \in E$. Now, consider the graph H resulting from removing v and its neighbors from G. Clearly, $\gamma(H)$ is larger (or equal) to the total charge of the vertices of $V(H)$ in G, as their degree had either decreased (or remained the same). As such, by induction, we have an independent set in H of size at least $\gamma(H)$. Together with v this forms an independent set in G of size at least $\gamma(H)+1 \geq \gamma(G)$. Implying that there exists an independent set in G of size

$$\tau = \sum_{v \in V} \frac{1}{1+d(v)}, \tag{15.2}$$

Now, set $x_v = 1 + d(v)$, and observe that

$$(n+2|E|)\tau = \left(\sum_{v \in V} x_v\right)\left(\sum_{v \in V} \frac{1}{x_v}\right) \geq \sum_{v \in V} x_v \frac{1}{x_v} = n.$$

Namely, $\tau \geq \dfrac{n}{n+2|E|} = \dfrac{1}{1+2|E|/n} = \dfrac{1}{1+d_G}$. ∎

15.0.2. An algorithm for the weighted case

In the weighted case, we associate weight $w(v)$ with each vertex of G, and we are interested in the maximum weight independent set in G. Deploying the algorithm described in the first proof of Theorem 15.0.1, implies the following.

Lemma 15.0.2. *The graph $G = (V, E)$ has an independent set of size $\geq \sum_{v \in V} \dfrac{w(v)}{1+d(v)}$.*

Proof: By linearity of expectations, we have that the expected weight of the independent set computed is equal to

$$\sum_{v \in V} w(v) \cdot \mathbb{P}[v \text{ in the independent set}] = \sum_{v \in V} \frac{w(v)}{1+d(v)}, \qquad \blacksquare$$

Chapter 16

Frequency Estimation over a Stream

"See? Genuine-sounding indignation. I programmed that myself. It's the first thing you need in a university environment: the ability to take offense at any slight, real or imagined."

Robert Sawyer, Factoring Humanity

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
December 10, 2019

16.1. Frequency estimation over a stream for the kth moment

Let $\mathcal{S} = (s_1, \ldots, s_m)$ be a sequence of elements from $N = \{1, \ldots, n\}$. Let f_i be the number of times the number i appears in $\mathcal{S}$. For $k \geq 0$, let

$$F_k = \sum_{i=1}^{n} f_i^k$$

be the ***kth frequency moment*** of $\mathcal{S}$. The quantity, $F_1 = m$ is the length of the stream $\mathcal{S}$. Similarly, F_0 is the number of distinct elements (where we use the convention that $0^0 = 0$ and any other quantity to the power 0 is 1). It is natural to define $F_\infty = \max_i f_i$.

Here, we are interested in approximating up to a factor of $1 \pm \varepsilon$ the quantity F_k, for $k \geq 1$ using small space, and reading the stream $\mathcal{S}$ only once.

16.1.1. An estimator

16.1.1.1. Computing the estimate

One can pick a representative element from a stream uniformly at random by using reservoir sampling. That is, sample the ith element s_i to be the representative with probability $1/i$. Once sampled, the algorithm counts how many times it see the representative value later on in the stream (the counter is initialized to 1, to account for the chosen representative itself). In particular, if s_p is the chosen representative in the end of the stream (i.e., the algorithm might change the representative several times), then the counter r is the size of the set

$$\left\{ j \;\middle|\; j \geq p \text{ and } s_j = s_p \right\}.$$

the output of the algorithm is the quantity

$$X = m\big(r^k - (r-1)^k\big).$$

Let V be the random variable that is the value of the representative in the end of the sequence (i.e., V).

16.1.1.2. Analysis

Lemma 16.1.1. *We have* $\mathbb{E}[X] = F_k$.

Proof: Observe that since we choose the representative uniformly at random, we have

$$\mathbb{E}[X \mid V = i] = \sum_{j=1}^{f_i} \frac{1}{f_i} m\big(j^k - (j-1)^k\big) = \frac{m}{f_i} \sum_{j=1}^{f_i} \big(j^k - (j-1)^k\big) = \frac{m}{f_i} f^k.$$

As such, we have $\mathbb{E}[X] = \mathbb{E}\big[\mathbb{E}[X \mid V]\big] = \sum_{i:f_i \neq 0} \frac{f_i}{m} \frac{m}{f_i} f_i^k = \sum_i f_i^k = F_k$. ■

Remark 16.1.2. In the above, we estimated the function $g(x) = x^k$, over the frequency numbers $f_1, \ldots, f_k$, but the above argumentation, on the expectation of X, would work for any function $g(x)$ such that $g(0) = 0$, and $g(x) \geq 0$, for all $x \geq 0$.

Lemma 16.1.3. *For* $k > 1$, *we have* $\sum_{i=1}^n \big(i^k - (i-1)^k\big)^2 \leq k n^{2k-1}$.

Proof: Observe that for $x \geq 1$, we have that $x^k - (x-1)^k \leq k x^{k-1}$. As such, we have

$$\sum_{i=1}^n \Big(i^k - (i-1)^k\Big)^2 \leq \sum_{i=1}^n k i^{k-1} \Big(i^k - (i-1)^k\Big) \leq k n^{k-1} \sum_{i=1}^n \Big(i^k - (i-1)^k\Big) = k n^{k-1} n^k = k n^{2k-1}.$$ ■

Lemma 16.1.4. *We have* $\mathbb{E}\big[X^2\big] \leq k m F_{2k-1}$.

Proof: By Lemma 16.1.3, we have $\mathbb{E}\big[X^2 \,\big|\, V = i\big] = \sum_{j=1}^{f_i} \frac{1}{f_i} m^2 \big(j^k - (j-1)^k\big)^2 \leq \frac{m^2}{f_i} k f_i^{2k-1} = m^2 k f_i^{2k-2}$, and thus $\mathbb{E}\big[X^2\big] = \mathbb{E}\big[\mathbb{E}\big[X^2 \,\big|\, V\big]\big] = \sum_{i:f_i \neq 0} \frac{f_i}{m} \cdot m^2 k f_i^{2k-2} = m k F_{2k-1}$. ■

Lemma 16.1.5. *For any non-negative numbers* $f_1, \ldots, f_n$, *and* $k \geq 1$, *we have*

$$\sum_{i=1}^n f_i \leq n^{(k-1)/k} \Big(\sum_{i=1}^n f_i^k\Big)^{1/k}.$$

Proof: The above is equivalent to proving that $\sum_i f_i/n \leq \left(\sum_{i=1}^n f_i^k/n\right)^{1/k}$. Raising both sides to the power k, we need to show that $\big(\sum_i f_i/n\big)^k \leq \sum_{i=1}^n f_i^k/n$. Setting $g(x) = x^k$, we have $g\big(\sum_i f_i/n\big) \leq \sum_{i=1}^n g(f_i)/n$. The last inequality holds by the convexity of the function $g(x)$ (indeed, $g'(x) = k x^{k-1}$ and $g''(x) = k(k-1)x^{k-2} \geq 0$, for $x \geq 0$). ■

Lemma 16.1.6. *For any* n *numbers* $f_1, \ldots, f_n \geq 0$, *we have* $\Big(\sum_i f_i\Big)\Big(\sum_i f_i^{2k-1}\Big) \leq n^{1-1/k} \Big(\sum_i f_i^k\Big)^2$.

Proof: Let $M = \max_i f_i$. We have

$$\sum_i f_i^{2k-1} \le M^{k-1} \sum_i f_i^k \le M^{k(k-1)/k} \sum_i f_i^k \le \left(\sum_i f_i^k\right)^{(k-1)/k} \sum_i f_i^k \le \left(\sum_i f_i^k\right)^{(2k-1)/k}.$$

By Lemma 16.1.5, we have $\sum_{i=1}^n f_i \le n^{(k-1)/k}\left(\sum_i f_i^k\right)^{1/k}$. Multiplying the above two inequality implies the claim. ∎

Lemma 16.1.7. *We have* $\mathbb{V}[X] \le kn^{1-1/k}F_k^2$.

Proof: Since $m = \sum_i f_i$, Lemma 16.1.4 and Lemma 16.1.6 together implies that $\mathbb{V}[X] = \mathbb{E}\left[X^2\right] - (\mathbb{E}[X])^2 \le \mathbb{E}\left[X^2\right] \le kmF_{2k-2} \le kn^{1-1/k}F_k^2$. ∎

16.1.2. An improved estimator

Let $\varphi, \varepsilon \in (0,1)$ be parameters. In the following, let

$$\alpha = 8kn^{1-1/k}/\varepsilon^2 \qquad \text{and} \qquad \beta = 4\ln\frac{1}{\varphi}.$$

Let us use α copies of the above estimator (running in parallel), and then take their average, which results in a new estimator. Let repeat this β times, as such, we get β "average" estimators $Y_1, \ldots, Y_\beta$ (i.e., there are $\alpha\beta$ independent copies of the simple estimator being executed in parallel). Let Z be the median value of $Y_1, \ldots, Y_\beta$, and we output Z as the new estimate.

16.1.2.1. Analysis

Lemma 16.1.8. *For* $i = 1, \ldots, j$, *we have* $\mathbb{P}[|Y_i - F_k| > \varepsilon F_k] \le \frac{1}{8}$.

Proof: The variable $X_{i,j}$ is a basic estimator being computed, for $i = 1, \ldots, \beta$, and $j = 1, \ldots, \alpha$. The variable $Y_i = \sum_{j=1}^{\alpha} X_{i,j}/\alpha$, for all i. Since all the $X_{i,j}$s are independent, and have the same distribution as X, we have

$$\mathbb{V}\left[Y_i\right] = \mathbb{V}\left[\frac{1}{\alpha}\sum_{j=1}^{\alpha} X_{i,j}\right] = \frac{1}{\alpha^2}\mathbb{V}\left[\sum_{j=1}^{\alpha} X_{i,j}\right] = \frac{1}{\alpha^2}\sum_{j=1}^{\alpha}\mathbb{V}\left[X_{i,j}\right] = \frac{1}{\alpha^2}\alpha\,\mathbb{V}[X] = \frac{\mathbb{V}[X]}{\alpha}.$$

Similarly, we have $\mathbb{E}[Y_i] = \mathbb{E}\left[\sum_{j=1}^{\alpha} X_{i,j}/k\right] = \mathbb{E}[X]$. Namely, the effect of averaging α independent copies of the same variable is to reduce the variance by a factor of α, while keeping the expectation the same.

Let $t = \varepsilon F_k/\sigma_i$, where $\sigma_i = \sqrt{\mathbb{V}[Y_i]}$. By Chebychev's inequality (Theorem 1.3.4) and Lemma 16.1.7, we have

$$\mathbb{P}[|Y_i - F_k| > \varepsilon F_k] = \mathbb{P}[|Y_i - F_k| > t\sigma_i] \le \frac{1}{t^2} = \frac{\sigma_i^2}{\varepsilon^2 F_k^2} = \frac{\mathbb{V}[Y_i]}{\varepsilon^2 F_k^2} = \frac{\mathbb{V}[X]}{\alpha}\cdot\frac{1}{\varepsilon^2 F_k^2} \le \frac{kn^{1-1/k}F_k^2}{\alpha\varepsilon^2 F_k^2}.$$

We want the last quantity to be smaller than $1/8$, which requires that $\frac{kn^{1-1/k}}{\alpha\varepsilon^2} \le 1/8$, which holds for $\alpha = 8kn^{1-1/k}/\varepsilon^2$, which is (surprise, surprise) the value assigned to α. ∎

Lemma 16.1.9. *We have for the estimator* Z *that* $\mathbb{P}[|Z - F_k| \ge \varepsilon F_k] \le \varphi$.

Proof: The probability of an estimator Y_i to be bad, for $i = 1, \ldots, \beta$, is at most $1/8$. Let U be the random variable that is the number of estimators that are bad. The variable U has binomial distribution with β coin tosses and probability $1/8$. As such, we have

$$\mathbb{P}[Z \text{ is bad}] \leq \mathbb{P}[U \geq \beta/2] = \mathbb{P}[U \geq (1+3)\beta/8] \leq \exp(-(\beta/8)3^2/4) \leq \exp\left(-\ln \frac{1}{\varphi}\right) = \varphi,$$

by Chernoff inequality (Lemma 8.2.5), and plugging in the value of $\beta = 4 \ln \frac{1}{\varphi}$. ■

In the following, we consider a computer ***word*** to be sufficiently large to contain $\lg n$ or $\lg m$ bits. This readily implies the following.

Theorem 16.1.10. *Let $\mathcal{S} = (strm_1, \ldots, s_n)$ be a stream of numbers from the set $\{1, \ldots, n\}$. Let $k \geq 1$ be a parameter. Given $\varepsilon, \varphi \in (0,1)$, one can build a data-structure using $O(kn^{1-1/k}\varepsilon^{-2} \log \varphi^{-1})$ words, such that one can $(1 \pm \varepsilon)$-approximate the kth moment of the elements in the stream; that is, the algorithm outs a number Z, such that $(1-\varepsilon)F_k \leq Z \leq (1+\varepsilon)F_k$, where $F_k = \sum_{i=1}^{n} f_i^k$, and f_i is the number of times i appears in the stream $\mathcal{S}$. The algorithm succeeds with probability $\geq 1 - \varphi$.*

16.2. Better estimation for F_2

16.2.1. Pseudo-random k-wide independent sequence of signed bits

In the following, assume that we sample $O(\log n)$ bits, such that given an index i, one can compute (quickly!) a random signed bit $b(i) \in \{-1, +1\}$. We require that the resulting bits $b(1), b(2), \ldots, b(n)$ are 4-wise independent. To this end, pick a prime p, that is, say bigger than n^{10}. This can easily be done by sampling a number in the range $[n^{10}, n^{11}]$, and checking if it is prime (which can done in polynomial time).

Once we have such a prime, we generate a random polynomial $g(i) = \sum_{i=0}^{5} c_i x^i \bmod p$, by choosing $c_0, \ldots, c_5$ from $\mathbb{Z}_p = \{0, \ldots, p-1\}$. We had seen that $g(0), g(1), \ldots, g(n)$ are uniformly distributed in $\mathbb{Z}_p$, and they are, say, 6-wise independent (see Theorem 3.2.9).

We define

$$b(i) = \begin{cases} 0 & g(i) = p-1 \\ +1 & g(i) \text{ is odd} \\ -1 & g(i) \text{ is even.} \end{cases}$$

Clearly, the sequence $b(1), \ldots, b(n)$ are 6-wise independent. There is a chance that one of these bits might be zero, but the probability for that is at most n/p, which is so small, that we just assume it does not happen. There are known constructions that do not have this issue at all (one of the bits is zero), but they are more complicated.

Lemma 16.2.1. *Given a parameter $\varphi \in (0,1)$, in polynomial time in $O(\log(n/\varphi))$, one can construct a function $b(\cdot)$, requiring $O(\log(n/BadProb))$ bits of storage (or $O(1)$ words), such that $b(1), \ldots, b(n) \in \{-1, +1\}$ with equal probability, an they are 6-wise independent. Furthermore, given i, one can compute $b(i)$ in $O(1)$ time.*

The probability of this sequence to fail having the desired properties is smaller than φ.

Proof: We repeat the above construction, but picking a prime p in the range, say, $n^{10}/\varphi \ldots n^{11}/\varphi$. ■

16.2.2. Estimator construction for F_2

16.2.2.1. The basic estimator

As before we have the stream $\mathcal{S} = s_1 \ldots, s_m$ of numbers from the set $1, \ldots, n$. We compute the 6-wise independent sequence of random bits of Lemma 16.2.1, and in the following we assume this sequence is good (i.e., has only -1 and $+1$ in it). We compute the quantity

$$T = \sum_{i=1}^{m} b(i) f_i = \sum_{j=1}^{m} b(s_j),$$

which can be computed on the fly using $O(1)$ words of memory, and $O(1)$ time per time in the stream.

The algorithm returns $X = T^2$ as the desired estimate.

Analysis.

Lemma 16.2.2. *We have* $\mathbb{E}[X] = \sum_i f_i^2 = F_2$ *and* $\mathbb{V}[X] \le 2F_2^2$.

Proof: We have that $\mathbb{E}[X] = \mathbb{E}\left[\left(\sum_{i=1}^{n} b(i) f_i\right)^2\right]$, and as such

$$\mathbb{E}[X] = \mathbb{E}\left[\sum_{i=1}^{n} (b(i))^2 f_i^2 + 2\sum_{i<j} b(i)b(j) f_i f_j\right] = \sum_{i=1}^{m} f_i^2 + 2\sum_{i<j} f_i f_j \,\mathbb{E}[b(i)b(j)] = \sum_{i=1}^{m} f_i^2 = F_2,$$

since $\mathbb{E}[b(i)] = 0$, $\mathbb{E}\left[b(i)^2\right] = 1$, and $\mathbb{E}[b(i)b(j)] = \mathbb{E}[b(i)]\,\mathbb{E}[b(j)] = 0$ (assuming the sequence $b(1), \ldots, b(n)$ has not failed), by the 6-wise Independence of the sequence of signed bits.

We next computer $\mathbb{E}\left[X^2\right]$. To this end, let $N = \{1, \ldots, n$, and $\Gamma = N \times N \times N \times N$. We split this set into several sets, as follows:

(i) $\Gamma_0 = \left\{(i,i,i,i) \in N^4\right\}$: All quadruples that are all the same value.

(ii) Γ_1: Set of all quadruples (i,j,k,l) where there is at least one value that appears exactly once.

(iii) Γ_2: Set of all (i,j,k,ℓ) with only two distinct values, each appearing exactly twice.

Clearly, we have $N^4 = \Gamma_0 \cup \Gamma_1 \cup \Gamma_2$.

For a tuple $(i,i,i,i) \in \Gamma_0$, we have $\mathbb{E}[b(i)b(i)b(i)b(i)] = \mathbb{E}\left[b(i)^4\right] = 1$.

For a tuple $(i,j,k,\ell) \in \Gamma_1$ with i being the unique value, we have that

$$\mathbb{E}[b(i)b(j)b(k)b(\ell)] = \mathbb{E}[b(i)]\,\mathbb{E}[b(j)b(k)b(\ell)] = 0\,\mathbb{E}[b(j)b(k)b(\ell)] = 0,$$

using that the signed bits are 4-wise independent. The same argumentation implies that $\mathbb{E}[b(i)b(j)b(k)b(\ell)] = 0$ for any tuple $(i,j,k,\ell) \in \Gamma_1$.

For a tuple $(i,i,j,j) \in \Gamma_2$, we have $\mathbb{E}[b(i)b(i)b(j)b(j)] = \mathbb{E}\left[b(i)^2 b(j)^2\right] = \mathbb{E}\left[b(i)^2\right]\mathbb{E}\left[b(j)^2\right] = 1$, and the same argumentation applies to any tuple of Γ_2. Observe that for any $i < j$, there are $\binom{4}{2} = 6$ different tuples in Γ_2 that are made out of i and j. As such, we have

$$\begin{aligned}
\mathbb{E}\left[X^2\right] &= \mathbb{E}\left[\left(\sum_{i=1}^{n} b(i) f_i\right)^4\right] = \mathbb{E}\left[\sum_{(i,j,k,\ell)\in\Gamma} b(i)b(j)b(k)b(\ell) f_i f_j f_k f_\ell\right] \\
&= \sum_{(i,i,i,i)\in\Gamma_0} \mathbb{E}\left[b(i)^4\right] f_i^4 + \sum_{(i,j,k,\ell)\in\Gamma_1} f_i f_j f_k f_\ell \,\mathbb{E}[b(i)b(j)b(k)b(\ell)] + 6\sum_{i<j} \mathbb{E}\left[b(i)^2 b(j)^2\right] f_i^2 f_j^2 \\
&= \sum_{i=1}^{n} f_i^4 + 6\sum_{i<j} f_i^2 f_j^2
\end{aligned}$$

As such, we have

$$\mathbb{V}[X] = \mathbb{E}\left[X^2\right] - (\mathbb{E}[X])^2 = \sum_{i=1}^{n} f_i^4 + 6\sum_{i<j} f_i^2 f_j^2 - \left(\sum_{i=1}^{m} f_i^2\right)^2 = 4\sum_{i<j} f_i^2 f_j^2 \leq 2F_2^2. \qquad \blacksquare$$

16.2.3. Improving the estimator

We repeat the same scheme as above. Let $\varphi, \varepsilon \in (0,1)$ be parameters. In the following, let

$$\alpha = 16/\varepsilon^2 \qquad \text{and} \qquad \beta = 4\ln\frac{1}{\varphi}.$$

Let $X_{i,j}$ be a basic estimator for F_2, using the estimator of Section 16.2.2.1, for $i = 1, \ldots, \beta$ and $j = 1, \ldots, \alpha$. Let $Y_i = \sum_{j=1}^{\alpha} X_{i,j}/\alpha$, for $i = 1, \ldots, \beta$. Let Z be the median of $Y_1, \ldots, Y_\beta$, and the algorithm returns Z as the estimator.

Theorem 16.2.3. *Given a stream $\mathcal{S} = s_1, \ldots, s_m$ of numbers from $\{1, \ldots, n\}$, and parameters $\varepsilon, \varphi \in (0,1)$, one can compute an estimate Z for $F_2(\mathcal{S})$, such that $\mathbb{P}[|Z - F_2| > \varepsilon F_2] \leq \varphi$. This algorithm requires $O(\varepsilon^{-2}\log\varphi^{-1})$ space (in words), and this is also the time to handle a new element in the stream.*

Proof: The scheme is described above. As before, using Chebychev's inequality, we have that

$$\mathbb{P}[|Y_i - F_2| > \varepsilon F_2] = \mathbb{P}\left[|Y_i - F_2| > \frac{\varepsilon F_2}{\sqrt{\mathbb{V}[Y_i]}}\sqrt{\mathbb{V}[Y_i]}\right] \leq \frac{\mathbb{V}[Y_i]}{\varepsilon^2 F_2^2} = \frac{\mathbb{V}[X]/\alpha}{\varepsilon^2 F_2^2} \leq \frac{2F_2^2}{\alpha\varepsilon^2 F_2^2} = \frac{1}{8},$$

by Lemma 16.2.2. Let U be the number of estimators in $Y_1, \ldots, Y_\beta$ that are outside the acceptable range. Arguing as in Lemma 16.1.9, we have

$$\mathbb{P}[Z \text{ is bad}] \leq \mathbb{P}[U \geq \beta/2] = \mathbb{P}[U \geq (1+3)\beta/8] \leq \exp(-(\beta/8)3^2/4) \leq \exp\left(-\ln\frac{1}{\varphi}\right) = \varphi,$$

by Chernoff inequality (Lemma 8.2.5), and $\blacksquare$

16.3. Bibliographical notes

The beautiful results of this chapter are from a paper from Alon *et al.* [AMS99].

Chapter 17

Approximating the Number of Distinct Elements in a Stream

"See? Genuine-sounding indignation. I programmed that myself. It's the first thing you need in a university environment: the ability to take offense at any slight, real or imagined."

Robert Sawyer, Factoring Humanity

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
December 10, 2019

17.1. Counting number of distinct elements

17.1.1. First order statistic

Let $X_1, \ldots, X_u$ be u random variables uniformly distributed in $[0,1]$. Let $Y = \min(X_1, \ldots, X_u)$. The value Y is the ***first order statistic*** of $X_1, \ldots, X_u$.

Lemma 17.1.1. *We have* $\mathbb{E}[Y] = \frac{1}{u+1}$, $\mathbb{E}\left[Y^2\right] = \frac{2}{(u+1)(u+2)}$, *and* $\mathbb{V}[Y] = \frac{u}{(u+1)^2(u+2)}$.

Proof: Using integration by guessing, we have

$$\begin{aligned}\mathbb{E}[Y] &= \int_{y=0}^{1} y\,\mathbb{P}[Y=y]\,dy = \int_{y=0}^{1} y\cdot\binom{u}{1}1(1-y)^{u-1}\,dy = \int_{y=0}^{1} uy(1-y)^{u-1}\,dy \\ &= \left[-y(1-y)^u - \frac{(1-y)^{u+1}}{u+1}\right]_{y=0}^{1} = \frac{1}{u+1}.\end{aligned}$$

Using integration by guessing again, we have

$$\begin{aligned}\mathbb{E}\left[Y^2\right] &= \int_{y=0}^{1} y^2\,\mathbb{P}[Y=y]\,dy = \int_{y=0}^{1} y^2\cdot\binom{u}{1}1(1-y)^{u-1}\,dy = \int_{y=0}^{1} uy^2(1-y)^{u-1}\,dy \\ &= \left[-y^2(1-y)^u - \frac{2y(1-y)^{u+1}}{u+1} - \frac{2(1-y)^{u+2}}{(u+1)(u+2)}\right]_{y=0}^{1} = \frac{2}{(u+1)(u+2)}.\end{aligned}$$

We conclude that

$$\mathbb{V}[Y] = \mathbb{E}\left[X^2\right] - (\mathbb{E}[X])^2 = \frac{2}{(u+1)(u+2)} - \frac{1}{(u+1)^2} = \frac{1}{u+1}\left(\frac{2}{u+2} - \frac{1}{u+1}\right) = \frac{u}{(u+1)^2(u+2)}. \qquad \blacksquare$$

17.1.2. The algorithm

A single estimator. Assume that we have a perfectly random hash function h that randomly maps $N = \{1, \ldots, n\}$ to $[0, 1]$. Assume that the stream has u unique numbers in N. Then the set $\{h(s_1), \ldots, h(s_m)\}$ contains u random numbers uniformly distributed in $[0.1]$. The algorithm as such, would compute $X = \min_i h(s_i)$. Bu the above, we have $\mathbb{E}[X] = 1/(u+1)$, ans as such, $1/X - 1$ is a estimator for u.

A better estimator. We are going to compute $\alpha = 144/\varepsilon^2$ basic estimators: $X_1, \ldots, X_\alpha$. Let Y be the average of these variables. We have that $\mathbb{E}[Y] = 1/(u+1)$, and $\mathbb{V}[Y] = \mathbb{V}[X]/\alpha$. We require that

$$3\sqrt{\mathbb{V}[Y]} \leq \frac{\varepsilon}{4}\mathbb{E}[Y] \iff 12\sqrt{\frac{u}{\alpha(u+1)^2(u+2)}} \leq \varepsilon\frac{1}{u+1} \iff 12\sqrt{\frac{u}{\alpha(u+2)}} \leq \varepsilon$$

Now, using Chebychev's inequality, we have

$$\mathbb{P}\left[|Y - \mathbb{E}[Y]| > \frac{\varepsilon}{4}\mathbb{E}[Y]\right] \leq \mathbb{P}\left[|Y - \mathbb{E}[Y]| > 3\sqrt{\mathbb{V}[Y]}\right] \leq \frac{1}{9}.$$

Assume this bad event does not happen. Then we have that

$$\begin{aligned}
&(1-\varepsilon/4)\frac{1}{u+1} \leq Y \leq (1+\varepsilon/4)\frac{1}{u+1} \implies \frac{u+1}{1+\varepsilon/4} \leq \frac{1}{Y} \leq \frac{u+1}{1-\varepsilon/4} \implies \frac{u-\varepsilon/4}{1+\varepsilon/4} \leq \frac{1}{Y} - 1 \leq \frac{u+\varepsilon/4}{1-\varepsilon/4} \\
&\implies (1-\varepsilon/4)(u-\varepsilon/4) \leq \frac{1}{Y} - 1 \leq (1+\varepsilon/2)(u+\varepsilon/4) \\
&\implies (1-\varepsilon)u \leq \frac{1}{Y} - 1 \leq (1+\varepsilon)u.
\end{aligned}$$

This implies that $\frac{1}{Y} - 1$ is a $1 \pm \varepsilon$ approximation to the number of distinct values in the stream, with probability at least $8/9$.

A high probability estimator. Given a desired failure probability φ, the algorithm maintain $\beta = O(\log \varphi^{-1})$ independent copies of the better estimator. Using Chernoff's inequality, as we seen before, implies that the median of these estimators is the desired approximation with probability $\geq 1 - \varphi$.

Lemma 17.1.2. *Under the unreasonable assumption that we can sample perfectly random functions from $\{1, \ldots, n\}$ to $[0, 1]$, and storing such a function requires $O(1)$ words, then one can estimate the number of unique elements in a stream, using $O(\varepsilon^{-2} \log \varphi)$ words.*

17.2. Sampling from a stream with "low quality" randomness

Assume that we have a stream of elements $\mathcal{S} = s_1, \ldots, s_m$, all taken from the set $\{1, \ldots, n\}$, and we have a random sequence of bits $\mathcal{B} \equiv B_1, \ldots, B_n$, such that $\mathbb{P}[B_i = 1] = p$, for some p. Furthermore, we can compute B_i efficiently. Assume that the bits of $\mathcal{X}$ are pairwise independent. Let $F_0 = F_0(\mathcal{S})$ be the number of distinct values in the stream $\mathcal{S}$.

The sampling algorithm. When the ith arrives s_i, we compute B_{s_i}. If this bit is 1, then we insert s_i into the random sample R (if it is already in R, there is no need to store a second copy, naturally).

This defines a natural random sample

$$R = \{i \mid B_i = 1 \text{ and } i \in S\} \subseteq S.$$

Lemma 17.2.1. *For the above random sample R, let $X = |R|$. We have that $\mathbb{E}[X] = p\nu$ and $\mathbb{V}[X] = p\nu - p^2\nu$.*

Proof: Let $X = |R|$, and we have

$$\mathbb{E}[X] = \mathbb{E}\Big[\sum_{i\in S} B_i\Big] = \sum_{i\in S} \mathbb{E}[B_i] = p\nu.$$

$$\mathbb{E}\big[X^2\big] = \mathbb{E}\Big[(\sum_{i\in S} B_i)^2\Big] = \sum_{i\in S} \mathbb{E}\big[B_i^2\big] + 2\sum_{i,j\in S, i<j} \mathbb{E}\big[B_i B_j\big] = p\nu + 2\sum_{i,j\in S, i<j} \mathbb{E}[B_i]\,\mathbb{E}\big[B_j\big] = p\nu + 2p^2\binom{\nu}{2}.$$

As such, we have

$$\begin{aligned}\mathbb{V}[X] = \mathbb{V}[|R|] &= \mathbb{E}\big[X^2\big] - (\mathbb{E}[X])^2 = p\nu + 2p^2\binom{\nu}{2} - p^2\nu^2 = p\nu + 2p^2\frac{\nu(\nu-1)}{2} - p^2\nu^2 \\ &= p\nu + p^2\nu(\nu-1) - p^2\nu^2 = p\nu - p^2\nu.\end{aligned}$$

■

Lemma 17.2.2. *Let $\varepsilon \in (0, 1/4)$. Given $O(1/\varepsilon^2)$ space, and a parameter N, and the task is to estimate the size S, where we know that $|S| > N/4$. Then, there is an algorithm that would output one of the following:*

(A) $|S| > 2N$.

(B) Output a number ρ such that $(1-\varepsilon)|R| \le \rho \le (1+\varepsilon)|R|$.

(Note, that the two options are not disjoint.) The output of this algorithm is correct, with probability $\ge 7/8$.

Proof: We set $p = \frac{c}{N\varepsilon^2}$, where c is a constant to be determined shortly. Let $T = pN = O(1/\varepsilon^2)$. We sample a random sample R from S, by scanning the elements of S, and adding $i \in S$ to R if $B_i = 1$, If the random sample is larger than $8T$, at any point, then the algorithm outputs that $|S| > 2N$.

In all other cases, the algorithm outputs $|R|/p$ as the estimate for the size of S, together with R.

To bound the failure probability, consider first the case that $N/4 < |S|$. In this case, we have by the above, that

$$\mathbb{P}[|X - \mathbb{E}[X]| > \varepsilon\,\mathbb{E}[X]] \le \mathbb{P}\left[|X - \mathbb{E}[X]| > \varepsilon \frac{\mathbb{E}[X]}{\sqrt{\mathbb{V}[X]}}\sqrt{\mathbb{V}[X]}\right] \le \varepsilon^2 \frac{\mathbb{V}[X]}{(\mathbb{E}[X])^2} \le \frac{1}{8},$$

if $\frac{\mathbb{V}[X]}{\varepsilon^2(\mathbb{E}[X])^2} \le \frac{1}{8}$, For $\nu = |S| \ge N/4$, this happens if $\frac{p\nu}{\varepsilon^2 p^2\nu^2} \le \frac{1}{8}$. This in turn is equivalent to $8/\varepsilon^2 \le p\nu$. This is in turn happens if

$$\frac{c}{N\varepsilon^2} \cdot \frac{N}{4} \ge \frac{8}{\varepsilon^2},$$

which implies that this holds for $c = 32$. Namely, the algorithm in this case would output a $(1 \pm \varepsilon)$-estimate for $|S|$.

If the sample get bigger than $8T$, then the above readily implies that with probability at least $7/8$, the size of S is at least $(1-\varepsilon)8T/p > 2N$, Namely, the output of the algorithm is correct in this case.■

Lemma 17.2.3. *Let $\varepsilon \in (0, 1/4)$ and $\varphi \in (0, 1)$. Given $O(\varepsilon^{-2} \log \varphi^{-1})$ space, and a parameter N, and the task is to estimate F_0 of $\mathcal{S}$, given that $F_0 > N/4$. Then, there is an algorithm that would output one of the following:*
(A) $F_0 > 2N$.
(B) Output a number ρ such that $(1-\varepsilon)F_0 \leq \rho \leq (1+\varepsilon)F_0$.
(Note, that the two options are not disjoint.) The output of this algorithm is correct, with probability $\geq 1 - \varphi$.

Proof: We run $O(\log \varphi^{-1})$ copies of the of Lemma 17.2.2. If half of them returns that $F_0 > 2N$, then the algorithm returns that $F_0 > 2N$. Otherwise, the algorithm returns the median of the estimates returned, and return it as the desired estimated. The correctness readily follows by a repeated application of Chernoff's inequality. ■

Lemma 17.2.4. *Let $\varepsilon \in (0, 1/4)$. Given $O(\varepsilon^{-2} \log^2 n)$ space, one can read the stream $\mathcal{S}$ once, and output a number ρ, such that $(1-\varepsilon)F_0 \leq \rho \leq (1+\varepsilon)F_0$. The estimate is correct with high probability (i.e., $\geq 1 - 1/n^{O(1)}$).*

Proof: Let $N_i = 2^i$, for $i = 1, \ldots, M = \lceil \lg n \rceil$. Run M copies of Lemma 17.2.3, for each value of N_i, with $\varphi = 1/n^{O(1)}$. Let $Y_1, \ldots, Y_M$ be the outputs of these algorithms for the stream. A prefix of these outputs, are going to be "$F_0 > 2N_i$", Let j be the first Y_j that is a number. Return this number as the desired estimate. The correctness is easy – the first estimate that is a number, is a correct estimate with high probability. Since $N_M \geq n$, it also follows that Y_M must be a number. As such, there is a first number in the sequence, and the algorithm would output an estimate.

More precisely, there is an index i, such that $N_i/4 \leq F_0 \leq 2F_0$, and Y_i is a good estimate, with high probability. If any of the Y_j, for $j < i$, is an estimate, then it is correct (again) with high probability. ■

17.3. Bibliographical notes

Chapter 18

Approximate Nearest Neighbor (ANN) Search in High Dimensions

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
December 10, 2019

Possession of anything new or expensive only reflected a person's lack of theology and geometry; it could even cast doubts upon one's soul.

A confederacy of Dunces, John Kennedy Toole

18.1. ANN on the hypercube

18.1.1. ANN for the hypercube and the Hamming distance

Definition 18.1.1. The set of points $\mathcal{H}^d = \{0,1\}^d$ is the ***d-dimensional hypercube***. A point $\mathsf{p} = (\mathsf{p}_1, \ldots, \mathsf{p}_d) \in \mathcal{H}^d$ can be interpreted, naturally, as a binary string $\mathsf{p}_1\mathsf{p}_2\ldots\mathsf{p}_d$. The ***Hamming distance*** $\mathsf{d}_H(\mathsf{p},\mathsf{q})$ between $\mathsf{p},\mathsf{q} \in \mathcal{H}^d$ is the number of coordinates where p and q disagree.

It is easy to verify that the Hamming distance, being the L_1-norm, complies with the triangle inequality and is thus a metric.

As we saw previously, to solve the $(1+\varepsilon)$-ANN problem efficiently, it is sufficient to solve the ***approximate near neighbor*** problem. Namely, given a set P of n points in $\mathcal{H}^d$, a radius $r > 0$, and parameter $\varepsilon > 0$, we want to decide for a query point q whether $\mathsf{d}_H(\mathsf{q},\mathsf{P}) \le r$ or $\mathsf{d}_H(\mathsf{q},\mathsf{P}) \ge (1+\varepsilon)r$, where $\mathsf{d}_H(\mathsf{q},\mathsf{P}) = \min_{\mathsf{p}\in\mathsf{P}} \mathsf{d}_H(\mathsf{q},\mathsf{p})$.

Definition 18.1.2. For a set P of points, a data-structure $\mathsf{D} = \mathcal{D}_{\approx\mathrm{Near}}(\mathsf{P}, r, (1+\varepsilon)r)$ solves the ***approximate near neighbor*** problem if, given a query point q, the data-structure works as follows.

- NEAR: If $\mathsf{d}_H(\mathsf{q},\mathsf{P}) \le r$, then $\mathcal{D}$ outputs a point $\mathsf{p} \in \mathsf{P}$ such that $\mathsf{d}_H(\mathsf{p},\mathsf{q}) \le (1+\varepsilon)r$.
- FAR: If $\mathsf{d}_H(\mathsf{q},\mathsf{P}) \ge (1+\varepsilon)r$, then $\mathcal{D}$ outputs "$\mathsf{d}_H(\mathsf{q},\mathsf{P}) \ge r$".
- DON'T CARE: If $r \le \mathsf{d}(\mathsf{q},\mathsf{P}) \le (1+\varepsilon)r$, then $\mathcal{D}$ can return either of the above answers.

Given such a data-structure, one can construct a data-structure that answers the approximate nearest neighbor query using $O\big(\log\big(\varepsilon^{-1}\log d\big)\big)$ queries using an approximate near neighbor data-structure. Indeed, the desired distance $\mathsf{d}_H(\mathsf{q},\mathsf{P})$ is an integer number in the range $0, 1, \ldots, d$. We can build a $\mathcal{D}_{\approx\mathrm{Near}}$ data-structure for distances $(1+\varepsilon)^i$, for $i = 1, \ldots, M$, where $M = O\big(\varepsilon^{-1}\log d\big)$. Performing a binary search over these distances would resolve the approximate nearest neighbor query and requires $O(\log M)$ queries.

As such, in the following, we concentrate on constructing the approximate near neighbor data-structure (i.e., $\mathcal{D}_{\approx\mathrm{Near}}$).

18.1.2. Preliminaries

Definition 18.1.3. Consider a sequence m of k, not necessarily distinct, integers $i_1, i_2, \ldots, i_k \in [\![d]\!]$, where $[\![d]\!] = \{1, \ldots, d\}$. For a point $\mathsf{p} = (p_1, \ldots, p_d) \in \mathbb{R}^d$, its ***projection*** by m, denoted by $m\mathsf{p}$ is the point $(p_{i_1}, \ldots, p_{i_k}) \in \mathbb{R}^k$. Similarly, the *projection* of a point set $\mathsf{P} \subseteq \mathbb{R}^d$ by m is the point set $m\mathsf{P} = \{m\mathsf{p} \mid \mathsf{p} \in \mathsf{P}\}$.

Given two sequences $m = i_1, \ldots, i_k$ and $u = j_1, \ldots, j_{k'}$, let $m|u$ denote the *concatenated* sequence $m|u = i_1, \ldots, i_k, j_1, \ldots, j_{k'}$. Given a probability φ, a natural way to create such a projection, is to include the ith coordinate, for $i = 1, \ldots, d$, with probability φ. Let $\mathcal{D}_\varphi$ denote the distribution of such sequences of indices.

Definition 18.1.4. Let $\mathcal{D}_\varphi^T$ denote the distribution resulting from concatenating T independent sequences sampled from $\mathcal{D}_\varphi$. The length of a sampled sequence is dT.

Observe that for a point $\mathsf{p} \in \{0,1\}^d$, and $M \in \mathcal{D}_\varphi^T$, the projection $M\mathsf{p}$ might be higher dimensional than the original point p, as it might contain repeated coordinates of the original point.

18.1.2.1. Algorithm

18.1.2.1.1. Input. The input is a set P of n points in the hypercube $\{0,1\}^d$, and parameters r and ε.

18.1.2.1.2. Preprocessing. We set parameters as follows:

$$\beta = \frac{1}{1+\varepsilon} \in (0,1), \quad \varphi = 1 - \exp\left(-\frac{1}{r}\right) \approx \frac{1}{r}, \quad T = \beta \ln n, \quad \text{and} \quad L = O(n^\beta \log n).$$

We randomly and independently pick L sequences $M_1, \ldots, M_L \in \mathcal{D}_\varphi^T$. Next, the algorithm computes the point sets $\mathsf{Q}_i = M_i\mathsf{P}_i$, for $i = 1, \ldots, L$, and stores them each in a hash table, denoted by D_i, for $i = 1, \ldots, L$.

18.1.2.1.3. Answering a query. Given a query point $\mathsf{q} \in \{0,1\}^d$, the algorithm computes $\mathsf{q}_i = M_i\mathsf{q}$, for $i = 1, \ldots, L$. From each D_i, the algorithm retrieves a list ℓ_i of all the points that collide with q_i. The algorithm scans the points in the lists $\ell_1, \ldots, \ell_L$. If any of these points is in Hamming distance smaller than $(1+\varepsilon)r$, the algorithm returns it as the desired near-neighbor (and stops). Otherwise, the algorithm returns that all the points in P are in distance at least r from q.

18.1.2.2. Analysis

Lemma 18.1.5. *Let K be a set of r marked/forbidden coordinates. The probability that a sequence $M = (m_1, \ldots, m_T)$ sampled from $\mathcal{D}_\varphi^T$ does not sample any of the coordinates of K is $1/n^\beta$. This probability increases if K contains fewer coordinates.*

Proof: For any i, the probability that m_i does not contain any of these coordinates is $(1-\varphi)^r = \left(e^{-1/r}\right)^r = 1/e$. Since this experiment is repeated T times, the probability is $e^{-T} = e^{-\beta \ln n} = n^{-\beta}$. ∎

Lemma 18.1.6. *Let p be the nearest-neighbor to q in P. If $\mathsf{d}_H(\mathsf{q}, \mathsf{p}) \le r$ then, with high probability, the data-structure returns a point that is in distance $\le (1+\varepsilon)r$ from q.*

Proof: The good event here is that p and q collide under one of the sequences of $M_1, \ldots, M_L$. However, the probability that $M_i\mathsf{p} = M_i\mathsf{q}$ is at least $1/n^\beta$, by Lemma 18.1.5, as this is the probability that M_i does not sample any of the (at most r) coordinates where p and q disagree. As such, the probability that all L data-structures fail (i.e., none of the lists $\ell_1, \ldots, \ell_L$ contains p), is at most $(1 - 1/n^\beta)^L < 1/n^{O(1)}$, as $L = O\left(n^\beta \log n\right)$. ∎

Lemma 18.1.7. *In expectation, the total number of points in $\ell_1, \ldots, \ell_L$ that are in distance $\geq (1+\varepsilon)r$ from q is $\leq L$.*

Proof: Let $\mathsf{P}_{\geq}$ be the set of points in P that are in distance $\geq (1+\varepsilon)r$ from q. For a point $\mathsf{q} \in \mathsf{P}_{\geq}$, with $\Delta = \mathsf{d}_H(\mathsf{q}, \mathsf{q})$, we have that the probability for $M \in \mathcal{D}_\varphi^T$ misses all the Δ coordinates, where q and q differ, is

$$(1-\varphi)^{\Delta T} \leq (1-\varphi)^{(1+\varepsilon)rT} = \left(e^{-1/r}\right)^{(1+\varepsilon)rT} = \exp(-(1+\varepsilon)\beta \ln n) = \frac{1}{n},$$

as $\varphi = 1 - e^{-1/r}$, $T = \beta \ln n$, and $\beta = 1/(1+\varepsilon)$. But then, for any i, we have

$$\mathbb{E}\left[|\ell_i|\right] = \sum_{\mathsf{p} \in \mathsf{P}_{\geq}} \mathbf{Pr}_{M_i}\left[M_i\mathsf{p} = M_i\mathsf{q}\right] \leq \left|\mathsf{P}_{\geq}\right| \frac{1}{n} \leq 1.$$

As such, the total number of far points in the lists is at most $L \cdot 1 = L$, implying the claim. ∎

18.1.2.3. Running time

For each i, the query computes $M_i\mathsf{q}$ and that takes $O(dT) = O(d \log n)$ time. Repeated L times, this takes $O(Ld \log n)$ time overall. Let X be the random variable that is the number of points in the extracted lists that are in distance $\geq (1+\varepsilon)r$ from the query point. The time to scan the lists is $O\big(d(X+1)\big)$, since the algorithm stops as soon as it finds a near point. As such, by Lemma 18.1.7, the expected query time is $O(Ld \log n + Ld) = O\left(dn^{1/(1+\varepsilon)} \log^2 n\right)$.

18.1.2.3.1. Improving the performance (a bit). Observe that for $M \in \mathcal{D}_\varphi^T$, and any two points $\mathsf{p}, \mathsf{q} \in \{0,1\}^d$, all the algorithm cares about is whether $M\mathsf{p} = M\mathsf{q}$. As such, if a coordinate is probed many times by M, we might as well probe this coordinate only once. In particular, for a sequence $M \in \mathcal{D}_\varphi^T$, let $M' = \mathrm{uniq}(M)$ be the projection sequence resulting from removing replications in M. Significantly, M' is only of length $\leq d$, and as such, computing $M'\mathsf{p}$, for a point p, takes only $O(d)$ time. It is not hard to verify that one can also sample directly $\mathrm{uniq}(M)$, for $M \in \mathcal{D}_\varphi^T$, in $O(d)$ time. This improves the query and processing by a logarithmic factor.

We thus get the following result.

Theorem 18.1.8. *Given a set P of n points in $\{0,1\}^d$, and parameters r, ε, one can preprocess P in $O(dn^{1+1/(1+\varepsilon)} \log n)$ time and space, such that given a query point q, the algorithm returns, in expected $O(dn^{1/(1+\varepsilon)} \log n)$ time, one of the following:*

(A) a point $\mathsf{p} \in P$ such that $\mathsf{d}_H(\mathsf{q}, \mathsf{p}) \leq (1+\varepsilon)r$, or

(B) the distance of q from P is larger than r.

The algorithm may return either result if the distance of q from P is in the range $[r, (1+\varepsilon)r]$. The algorithm succeeds with high probability (per query).

One can also get a high-probability guarantee on the query time. For a parameter $\delta > 0$, create $O(\log \delta^{-1})$ LSH data-structures as above. Perform the query as above, except that when the query time exceeds (say) twice the expected time, move on to redo the query in the next LSH data-structure. The probability that the query had failed on one of these LSH data-structures is $\leq 1/2$, by Markov's inequality. As such, overall, the query time becomes $O(dn^{1/(1+\varepsilon)} \log n \log \delta^{-1})$, with probability $\geq 1 - \delta$.

18.2. LSH for the hypercube: An elaborate construction

We next present a similar scheme in a more systematic fashion – this would provide some intuition how we came up with the above construction.

18.2.0.1. On sense and sensitivity

Let $\mathsf{P} = \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}$ be a subset of vertices of the hypercube in d dimensions. In the following we assume that $d = n^{O(1)}$. Let $r, \varepsilon > 0$ be two prespecified parameters. We are interested in building an approximate near neighbor data-structure (i.e., $\mathcal{D}_{\approx \mathrm{Near}}$) for balls of radius r in the Hamming distance.

Definition 18.2.1. A family $\mathcal{F}$ of functions (defined over $\mathcal{H}^d$) is $\left(r, R, \widehat{\alpha}, \widehat{\beta}\right)$-***sensitive*** if for any $\mathsf{q}, \mathsf{r} \in \mathcal{H}^d$, we have the following

(A) If $\mathsf{q} \in \mathbf{b}(\mathsf{r}, r)$, then $\mathbb{P}[f(\mathsf{q}) = f(\mathsf{r})] \geq \widehat{\alpha}$.

(B) If $\mathsf{q} \notin \mathbf{b}(\mathsf{r}, R)$, then $\mathbb{P}[f(\mathsf{q}) = f(\mathsf{r})] \leq \widehat{\beta}$.

In (A) and (B), f is a randomly picked function from $\mathcal{F}$, $r < R$, and $\widehat{\alpha} > \widehat{\beta}$.

Intuitively, if we can construct an (r, R, α, β)-sensitive family, then we can distinguish between two points which are close together and two points which are far away from each other. Of course, the probabilities α and β might be very close to each other, and we need a way to do amplification.

A simple sensitive family.

A priori it is not even clear such a sensitive family exists, but it turns out that the family randomly exposing one coordinate is sensitive.

Lemma 18.2.2. *Let $f_i(\mathsf{p})$ denote the function that returns the ith coordinate of p, for $i = 1, \ldots, d$. Consider the family of functions $\mathcal{F} = \{f_1, \ldots, f_d\}$. Then, for any $r > 0$ and ε, the family $\mathcal{F}$ is $(r, (1+\varepsilon)r, \alpha, \beta)$-sensitive, where $\alpha = 1 - r/d$ and $\beta = 1 - r(1+\varepsilon)/d$.*

Proof: If $\mathsf{q}, \mathsf{r} \in \{0,1\}^d$ are within distance smaller than r from each other (under the Hamming distance), then they differ in at most r coordinates. The probability that a random $h \in \mathcal{F}$ would project into a coordinate that q and r agree on is $\geq 1 - r/d$.

Similarly, if $\mathsf{d}_H(\mathsf{q}, \mathsf{r}) \geq (1+\varepsilon)r$, then the probability that a random $h \in \mathcal{F}$ would map into a coordinate that q and r agree on is $\leq 1 - (1+\varepsilon)r/d$. ∎

A family with a large sensitivity gap.

Let k be a parameter to be specified shortly, and consider the family of functions $\mathcal{G}$ that concatenates k of the given functions. Formally, let

$$\mathcal{G} = \mathrm{combine}(\mathcal{F}, k) = \left\{ g \;\middle|\; g(\mathsf{p}) = \left(f^1(\mathsf{p}), \ldots, f^k(\mathsf{p})\right), \text{ for } f^1, \ldots, f^k \in \mathcal{F} \right\}$$

be the set of all such functions.

Lemma 18.2.3. *For a (r, R, α, β)-sensitive family $\mathcal{F}$, the family $\mathcal{G} = \text{combine}(\mathcal{F}, k)$ is $(r, R, \alpha^k, \beta^k)$-sensitive.*

Proof: For two fixed points $\mathsf{q}, \mathsf{r} \in \mathcal{H}^d$ such that $\mathsf{d}_H(\mathsf{q}, \mathsf{r}) \le r$, we have that for a random $h \in \mathcal{F}$, we have $\mathbb{P}[h(\mathsf{q}) = h(\mathsf{r})] \ge \alpha$. As such, for a random $g \in \mathcal{G}$, we have that

$$\begin{aligned}\mathbb{P}[g(\mathsf{q}) = g(\mathsf{r})] &= \mathbb{P}\big[f^1(\mathsf{q}) = f^1(\mathsf{r}) \text{ and } f^2(\mathsf{q}) = f^2(\mathsf{r}) \text{ and } \ldots \text{ and } f^k(\mathsf{q}) = f^k(\mathsf{r})\big] \\ &= \prod_{i=1}^{k} \mathbb{P}\big[f^i(\mathsf{q}) = f^i(\mathsf{r})\big] \ge \alpha^k.\end{aligned}$$

Similarly, if $\mathsf{d}_H(\mathsf{q}, \mathsf{r}) > R$, then $\mathbb{P}[g(\mathsf{q}) = g(\mathsf{r})] = \prod_{i=1}^{k} \mathbb{P}\big[f^i(\mathsf{q}) = f^i(\mathsf{r})\big] \le \beta^k$. ∎

The above lemma implies that we can build a family that has a gap between the lower and upper sensitivities; namely, $\alpha^k/\beta^k = (\alpha/\beta)^k$ is arbitrarily large. The problem is that if α^k is too small, then we will have to use too many functions to detect whether or not there is a point close to the query point.

Nevertheless, consider the task of building a data-structure that finds all the points of $\mathsf{P} = \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}$ that are equal, under a given function $g \in \mathcal{G} = \text{combine}(\mathcal{F}, k)$, to a query point. To this end, we compute the strings $g(\mathsf{p}_1), \ldots, g(\mathsf{p}_n)$ and store them (together with their associated point) in a hash table (or a prefix tree). Now, given a query point q, we compute $g(\mathsf{q})$ and fetch from this data-structure all the strings equal to it that are stored in it. Clearly, this is a simple and efficient data-structure. All the points colliding with q would be the natural candidates to be the nearest neighbor to q.

By not storing the points explicitly, but using a pointer to the original input set, we get the following easy result.

Lemma 18.2.4. *Given a function $g \in \mathcal{G} = \text{combine}(\mathcal{F}, k)$ (see Lemma 18.2.3) and a set $\mathsf{P} \subseteq \mathcal{H}^d$ of n points, one can construct a data-structure, in $O(nk)$ time and using $O(nk)$ additional space, such that given a query point q, one can report all the points in $X = \big\{\mathsf{p} \in \mathsf{P} \;\big|\; g(\mathsf{p}) = g(\mathsf{q})\big\}$ in $O(k + |X|)$ time.*

Amplifying sensitivity.

Our task is now to amplify the sensitive family we currently have. To this end, for two τ-dimensional points x and y, let $x \eqsim y$ be the Boolean function that returns *true* if there exists an index i such that $x_i = y_i$ and *false* otherwise. Now, the regular "=" operator requires vectors to be equal in all coordinates (i.e., it is equal to $\bigcap_i (x_i = y_i)$) while $x \eqsim y$ is $\bigcup_i (x_i = y_i)$. The previous construction of Lemma 18.2.3 using this alternative equal operator provides us with the required amplification.

Lemma 18.2.5. *Given an $\big(r, R, \alpha^k, \beta^k\big)$-sensitive family $\mathcal{G}$, the family $\mathcal{H}_{\eqsim} = \text{combine}(\mathcal{G}, \tau)$ if one uses the $\eqsim$ operator to check for equality is $\Big(r, R, 1 - \big(1 - \alpha^k\big)^\tau, 1 - \big(1 - \beta^k\big)^\tau\Big)$-sensitive.*

Proof: For two fixed points $\mathsf{q}, \mathsf{r} \in \mathcal{H}^d$ such that $\mathsf{d}_H(\mathsf{q}, \mathsf{r}) \le r$, we have, for a random $g \in \mathcal{G}$, that $\mathbb{P}[g(\mathsf{q}) = g(\mathsf{r})] \ge \alpha^k$. As such, for a random $h \in \mathcal{H}_{\eqsim}$, we have that

$$\begin{aligned}\mathbb{P}[h(\mathsf{q}) \eqsim h(\mathsf{r})] &= \mathbb{P}\big[g^1(\mathsf{q}) = g^1(\mathsf{r}) \text{ or } g^2(\mathsf{q}) = g^2(\mathsf{r}) \text{ or } \ldots \text{ or } g^\tau(\mathsf{q}) = g^\tau(\mathsf{r})\big] \\ &= 1 - \prod_{i=1}^{\tau} \mathbb{P}\big[g^i(\mathsf{q}) \neq g^i(\mathsf{r})\big] \ge 1 - \Big(1 - \alpha^k\Big)^\tau.\end{aligned}$$

Similarly, if $\mathsf{d}_H(\mathsf{q}, \mathsf{r}) > R$, then

$$\mathbb{P}[h(\mathsf{q}) \eqsim h(\mathsf{r})] = 1 - \prod_{i=1}^{\tau} \mathbb{P}\big[g^i(\mathsf{q}) \neq g^i(\mathsf{r})\big] \le 1 - \Big(1 - \beta^k\Big)^\tau.$$ ∎

To see the effect of Lemma 18.2.5, it is useful to play with a concrete example. Consider an $(r, R, \alpha^k, \beta^k)$-sensitive family where $\beta^k = \alpha^k/2$ and yet α^k is very small. Setting $\tau = 1/\alpha^k$, the resulting family is (roughly) $(r, R, 1 - 1/e, 1 - 1/\sqrt{e})$-sensitive. Namely, the gap shrank, but the threshold sensitivity is considerably higher. In particular, it is now a constant, and the gap is also a constant.

Using Lemma 18.2.5 as a data-structure to store P is more involved than before. Indeed, for a random function $h = \left(g^1, \ldots, g^\tau\right) \in \mathcal{H}_{\eqsim} = \text{combine}(\mathcal{G}, \tau)$ building the associated data-structure requires us to build τ data-structures for each one of the functions $g^1, \ldots, g^\tau$, using Lemma 18.2.4. Now, given a query point, we retrieve all the points of P that collide with each one of these functions, by querying each of these data-structures.

Lemma 18.2.6. *Given a function $h \in \mathcal{H}_{\eqsim} = \text{combine}(\mathcal{G}, \tau)$ (see Lemma 18.2.5) and a set $\mathsf{P} \subseteq \mathcal{H}^d$ of n points, one can construct a data-structure, in $O(nk\tau)$ time and using $O(nk\tau)$ additional space, such that given a query point q, one can report all the points in $X = \left\{\mathsf{p} \in \mathsf{P} \;\middle|\; h(\mathsf{p}) \eqsim h(\mathsf{q})\right\}$ in $O(k\tau + |X|)$ time.*

18.2.0.2. The near neighbor data-structure and handling a query

We construct the data-structure $\mathcal{D}$ of Lemma 18.2.6 with parameters k and τ to be determined shortly, for a random function $h \in \mathcal{H}_{\eqsim}$. Given a query point q, we retrieve all the points that collide with h and compute their distance to the query point. Next, scan these points one by one and compute their distance to q. As soon as encountering a point $\mathsf{r} \in \mathsf{P}$ such that $\mathsf{d}_H(\mathsf{q}, \mathsf{r}) \le R$, the data-structures returns ***true*** together with r.

Let's assume that we know that the expected number of points of $\mathsf{P} \setminus \mathbf{b}(\mathsf{q}, R)$ (i.e., $R = (1+\varepsilon)r$) that will collide with q in $\mathcal{D}$ is in expectation L (we will figure out the value of L below). To ensure the worst case query time, the query would abort after checking $4L + 1$ points and would return *false*. Naturally, the data-structure would also return false if all points encountered have distance larger than R from q.

Clearly, the query time of this data-structure is $O(k\tau + dL)$.

We are left with the task of fine-tuning the parameters τ and k to get the fastest possible query time, while the data-structure has reasonable probability to succeed. Figuring the right values is technically tedious, and we do it next.

18.2.0.3. Setting the parameters

If there exists $\mathsf{p} \in \mathsf{P}$ such that $\mathsf{d}_H(\mathsf{q}, \mathsf{p}) \le r$, then the probability of this point to collide with q under the function h is $\phi \ge 1 - \left(1 - \alpha^k\right)^\tau$. Let us demand that this data-structure succeeds with probability $\ge 3/4$. To this end, we set

$$\tau = 4\left\lceil 1/\alpha^k \right\rceil \implies \phi \ge 1 - \left(1 - \alpha^k\right)^\tau \ge 1 - \exp\left(-\alpha^k \tau\right) \ge 1 - \exp(-4) \ge 3/4, \tag{18.1}$$

since $1 - x \le \exp(-x)$, for $x \ge 0$.

Lemma 18.2.7. *The expected number of points of $\mathsf{P} \setminus \mathbf{b}(\mathsf{q}, R)$ colliding with the query point is $L = O\left(n(\beta/\alpha)^k\right)$, where $R = (1+\varepsilon)r$.*

Proof: Consider the points in $\mathsf{P} \setminus \mathbf{b}(\mathsf{q}, R)$. We would like to bound the number of points of this set that collide with the query point. Observe that in this case, the probability of a point $\mathsf{p} \in \mathsf{P} \setminus \mathbf{b}(\mathsf{q}, R)$ to

collide with the query point is

$$\leq \psi = 1 - \left(1-\beta^k\right)^\tau = \left(1-\left(1-\beta^k\right)\right)\left(1+\left(1-\beta^k\right)+\left(1-\beta^k\right)^2+\ldots+\left(1-\beta^k\right)^{\tau-1}\right)$$
$$\leq \beta^k \tau \leq 8\left(\frac{\beta}{\alpha}\right)^k,$$

as $\tau = 4\lceil 1/\alpha^k \rceil$ and $\alpha, \beta \in (0,1)$. Namely, the expected number of points of $\mathsf{P} \setminus \mathbf{b}(\mathsf{q}, R)$ colliding with the query point is $\leq \psi n$. ■

By Lemma 18.2.6, extracting the $O(L)$ points takes $O(k\tau + L)$ time. Computing the distance of the query time for each one of these points takes $O(k\tau + Ld)$ time. As such, by Lemma 18.2.7, the query time is

$$O(k\tau + Ld) = O\left(k\tau + nd(\beta/\alpha)^k\right).$$

To minimize this query time, we "approximately" solve the equation requiring the two terms, in the above bound, to be equal (we ignore d since, intuitively, it should be small compared to n). We get that

$$k\tau = n(\beta/\alpha)^k \rightsquigarrow \frac{k}{\alpha^k} \approx n\frac{\beta^k}{\alpha^k} \implies k \approx n\beta^k \rightsquigarrow 1/\beta^k \approx n \implies k \approx \ln_{1/\beta} n.$$

Thus, setting $k = \ln_{1/\beta} n$, we have that $\beta^k = 1/n$ and, by Eq. (18.1), that

$$\tau = 4\lceil 1/\alpha^k \rceil = \exp\left(\frac{\ln n}{\ln 1/\beta} \ln 1/\alpha\right) = O(n^\rho), \quad \text{for } \rho = \frac{\ln 1/\alpha}{\ln 1/\beta}. \tag{18.2}$$

As such, to minimize the query time, we need to minimize ρ.

Lemma 18.2.8. *(A) For $x \in [0,1)$ and $t \geq 1$ such that $1 - tx > 0$ we have $\dfrac{\ln(1-x)}{\ln(1-tx)} \leq \dfrac{1}{t}$.*

(B) For $\alpha = 1 - r/d$ and $\beta = 1 - r(1+\varepsilon)/d$, we have that $\rho = \dfrac{\ln 1/\alpha}{\ln 1/\beta} \leq \dfrac{1}{1+\varepsilon}$.

Proof: (A) Since $\ln(1-tx) < 0$, it follows that the claim is equivalent to $t\ln(1-x) \geq \ln(1-tx)$. This in turn is equivalent to

$$g(x) \equiv (1-tx) - (1-x)^t \leq 0.$$

This is trivially true for $x = 0$. Furthermore, taking the derivative, we see $g'(x) = -t + t(1-x)^{t-1}$, which is non-positive for $x \in [0,1)$ and $t > 0$. Therefore, g is non-increasing in the interval of interest, and so $g(x) \leq 0$ for all values in this interval.

(B) Indeed $\rho = \dfrac{\ln 1/\alpha}{\ln 1/\beta} = \dfrac{\ln \alpha}{\ln \beta} = \dfrac{\ln \frac{d-r}{d}}{\ln \frac{d-(1+\varepsilon)r}{d}} = \dfrac{\ln\left(1-\frac{r}{d}\right)}{\ln\left(1-(1+\varepsilon)\frac{r}{d}\right)} \leq \dfrac{1}{1+\varepsilon}$, by part (A). ■

In the following, it would be convenient to consider d to be considerably larger than r. This can be ensured by (conceptually) padding the points with fake coordinates that are all zero. It is easy to verify that this "hack" would not affect the algorithm's performance in any way and it is just a trick to make our analysis simpler. In particular, we assume that $d > 2(1+\varepsilon)r$.

Lemma 18.2.9. *For $\alpha = 1 - r/d$, $\beta = 1 - r(1+\varepsilon)/d$, n and d as above, we have that I. $\tau = O\left(n^{1/(1+\varepsilon)}\right)$, II. $k = O(\ln n)$, and III. $L = O\left(n^{1/(1+\varepsilon)}\right)$.*

Proof: By Eq. (18.1), $\tau = 4\lceil 1/\alpha^k \rceil = O(n^\rho) = O\left(n^{1/(1+\varepsilon)}\right)$, by Lemma 18.2.8(B).

Now, $\beta = 1 - r(1+\varepsilon)/d \leq 1/2$, since we assumed that $d > 2(1+\varepsilon)r$. As such, we have $k = \ln_{1/\beta} n = \dfrac{\ln n}{\ln 1/\beta} = O(\ln n)$.

By Lemma 18.2.7, $L = O\left(n(\beta/\alpha)^k\right)$. Now $\beta^k = 1/n$ and as such $L = O\left(1/\alpha^k\right) = O(\tau) = O\left(n^{1/(1+\varepsilon)}\right)$. ■

18.2.0.4. The result

Theorem 18.2.10. *Given a set P of n points on the hypercube $\mathcal{H}^d$ and parameters $\varepsilon > 0$ and $r > 0$, one can build a data-structure $\mathcal{D} = \mathcal{D}_{\approx Near}(\mathsf{P}, r, (1+\varepsilon)r)$ that solves the approximate near neighbor problem (see Definition 18.1.2). The data-structure answers a query successfully with high probability. In addition we have the following:*

(A) The query time is $O\left(dn^{1/(1+\varepsilon)} \log n\right)$.

(B) The preprocessing time to build this data-structure is $O\left(n^{1+1/(1+\varepsilon)} \log^2 n\right)$.

(C) The space required to store this data-structure is $O\left(nd + n^{1+1/(1+\varepsilon)} \log^2 n\right)$.

Proof: Our building block is the data-structure described above. By Markov's inequality, the probability that the algorithm has to abort because of too many collisions with points of $\mathsf{P} \setminus \mathbf{b}(\mathsf{q}, (1+\varepsilon)r)$ is bounded by $1/4$ (since the algorithm tries $4L+1$ points). Also, if there is a point inside $\mathbf{b}(\mathsf{q}, r)$, the algorithm would find it with probability $\geq 3/4$, by Eq. (18.1). As such, with probability at least $1/2$ this data-structure returns the correct answer in this case. By Lemma 18.2.6, the query time is $O(k\tau + Ld)$.

This data-structure succeeds only with constant probability. To achieve high probability, we construct $O(\log n)$ such data-structures and perform the near neighbor query in each one of them. As such, the query time is

$$O((k\tau + Ld)\log n) = O\left(n^{1/(1+\varepsilon)} \log^2 n + dn^{1/(1+\varepsilon)} \log n\right) = O\left(dn^{1/(1+\varepsilon)} \log n\right),$$

by Lemma 18.2.9 and since $d = \Omega(\lg n)$ if P contains n distinct points of $\mathcal{H}^d$.

As for the preprocessing time, by Lemma 18.2.6 and Lemma 18.2.9, we have

$$O(nk\tau \log n) = O\left(n^{1+1/(1+\varepsilon)} \log^2 n\right).$$

Finally, this data-structure requires $O(dn)$ space to store the input points. Specifically, by Lemma 18.2.6, we need an additional $O(nk\tau \log n) = O\left(n^{1+1/(1+\varepsilon)} \log^2 n\right)$ space. ■

In the hypercube case, when $d = n^{O(1)}$, we can build $M = O(\log_{1+\varepsilon} d) = O(\varepsilon^{-1} \log d)$ such data-structures such that $(1+\varepsilon)$-ANN can be answered using binary search on those data-structures which correspond to radii $r_1, \ldots, r_M$, where $r_i = (1+\varepsilon)^i$, for $i = 1, \ldots, M$.

Theorem 18.2.11. *Given a set P of n points on the hypercube $\mathcal{H}^d$ (where $d = n^{O(1)}$) and a parameter $\varepsilon > 0$, one can build a data-structure to answer approximate nearest neighbor queries (under the*

Hamming distance) using $O\left(dn + n^{1/(1+\varepsilon)}\varepsilon^{-1}\log^2 n \log d\right)$ *space, such that given a query point* q*, one can return a* $(1+\varepsilon)$*-ANN in* P *(under the Hamming distance) in* $O(dn^{1/(1+\varepsilon)}\log n \log(\varepsilon^{-1}\log d))$ *time. The result returned is correct with high probability.*

Remark 18.2.12. The result of Theorem 18.2.11 needs to be oblivious to the queries used. Indeed, for any instantiation of the data-structure of Theorem 18.2.11 there exist query points for which it would fail.

In particular, formally, if we perform a sequence of ANN queries using such a data-structure, where the queries depend on earlier returned answers, then the guarantee of a high probability of success is no longer implied by the above analysis (it might hold because of some other reasons, naturally).

18.3. LSH and ANN in Euclidean space

18.3.1. Preliminaries

Lemma 18.3.1. *Let* $X = (X_1, \ldots, X_d)$ *be a vector of* d *independent variables which have normal distribution* $\mathbf{N}$*, and let* $v = (v_1, \ldots, v_d) \in \mathbb{R}^d$*. We have that* $\langle v, X\rangle = \sum_i v_i X_i$ *is distributed as* $\|v\| Z$*, where* $Z \sim \mathbf{N}$*.*

Proof: By Lemma 13.2.3$_{\text{p103}}$ the point X has multi-dimensional normal distribution $\mathbf{N}^d$. As such, if $\|v\| = 1$, then this holds by the symmetry of the normal distribution. Indeed, let $e_1 = (1, 0, \ldots, 0)$. By the symmetry of the d-dimensional normal distribution, we have that $\langle v, X\rangle \sim \langle e_1, X\rangle = X_1 \sim \mathbf{N}$.

Otherwise, $\langle v, X\rangle / \|v\| \sim \mathbf{N}$, and as such $\langle v, X\rangle \sim N\left(0, \|v\|^2\right)$, which is indeed the distribution of $\|v\| Z$. ∎

Definition 18.3.2. A distribution $\mathcal{D}$ over $\mathbb{R}$ is called ***p-stable*** if there exists $p \geq 0$ such that for any n real numbers $v_1, \ldots, v_n$ and n independent variables $X_1, \ldots, X_n$ with distribution $\mathcal{D}$, the random variable $\sum_i v_i X_i$ has the same distribution as the variable $(\sum_i |v_i|^p)^{1/p} X$, where X is a random variable with distribution $\mathcal{D}$.

By Lemma 18.3.1, the normal distribution is a ***2-stable distribution***.

18.3.2. Locality sensitive hashing (LSH)

Let p and q be two points in $\mathbb{R}^d$. We want to perform an experiment to decide if $\|\mathsf{p} - \mathsf{q}\| \leq 1$ or $\|\mathsf{p} - \mathsf{q}\| \geq \eta$, where $\eta = 1 + \varepsilon$. We will randomly choose a vector $\mathbf{v}$ from the d-dimensional normal distribution $\mathbf{N}^d$ (which is 2-stable). Next, let r be a parameter, and let t be a random number chosen uniformly from the interval $[0, r]$. For $\mathsf{p} \in \mathbb{R}^d$, consider the random hash function

$$h(\mathsf{p}) = \left\lfloor \frac{\langle \mathsf{p}, \mathbf{v}\rangle + t}{r} \right\rfloor. \tag{18.3}$$

Assume that the distance between p and q is η and the distance between the projection of the two points to the direction $\mathbf{v}$ is β. Then, the probability that p and q get the same hash value is $\max(1 - \beta/r, 0)$, since this is the probability that the random sliding will not separate them. Indeed, consider the line through $\mathbf{v}$ to be the x-axis, and assume q is projected to r and r is projected to $r - \beta$

(assuming $r \geq \beta$). Clearly, q and r get mapped to the same value by $h(\cdot)$ if and only if $t \in [0, r-\beta]$, as claimed.

As such, we have that the probability of collusion is

$$\alpha(\eta, r) = \mathbb{P}[h(\mathsf{p}) = h(\mathsf{q})] = \int_{\beta=0}^{r} \mathbb{P}[|\langle \mathsf{p}, \mathbf{v} \rangle - \langle \mathsf{q}, \mathbf{v} \rangle| = \beta] \left(1 - \frac{\beta}{r}\right) d\beta.$$

However, since $\mathbf{v}$ is chosen from a 2-stable distribution, we have that $Z = \langle \mathsf{p}, \mathbf{v} \rangle - \langle \mathsf{q}, \mathbf{v} \rangle = \langle \mathsf{p} - \mathsf{q}, \mathbf{v} \rangle \sim \mathbf{N}\left(0, \|\mathsf{p} - \mathsf{q}\|^2\right)$. Since we are considering the absolute value of the variable, we need to multiply this by two. Thus, we have

$$\alpha(\eta, r) = \int_{\beta=0}^{r} \frac{2}{\sqrt{2\pi}\eta} \exp\left(-\frac{\beta^2}{2\eta^2}\right) \left(1 - \frac{\beta}{r}\right) d\beta,$$

by plugging in the density of the normal distribution for Z. Intuitively, we care about the difference $\alpha(1+\varepsilon, r) - \alpha(1, r)$, and we would like to maximize it as much as possible (by choosing the right value of r). Unfortunately, this integral is unfriendly, and we have to resort to numerical computation.

Now, we are going to use this hashing scheme for constructing locality sensitive hashing, as in the hypercube case, and as such we care about the ratio

$$\rho(1+\varepsilon) = \min_{r} \frac{\log(1/\alpha(1, r))}{\log(1/\alpha(1+\varepsilon, r))};$$

see Eq. (18.2)$_{\text{p133}}$.

The following is verified using numerical computations (using a computer).

Lemma 18.3.3 ([DIIM04]). *One can choose r, such that $\rho(1+\varepsilon) \leq \frac{1}{1+\varepsilon}$.*

Lemma 18.3.3 implies that the hash functions defined by Eq. (18.3) are $(1, 1+\varepsilon, \alpha', \beta')$-sensitive and, furthermore, $\rho = \frac{\log(1/\alpha')}{\log(1/\beta')} \leq \frac{1}{1+\varepsilon}$, for some values of α' and β'. As such, we can use this hashing family to construct an approximate near neighbor data-structure $\mathcal{D}_{\approx \mathrm{Near}}(\mathsf{P}, r, (1+\varepsilon)r)$ for the set P of points in $\mathbb{R}^d$. Following the same argumentation of Theorem 18.2.10, we have the following.

Theorem 18.3.4. *Given a set P of n points in $\mathbb{R}^d$ and parameters $\varepsilon > 0$ and $r > 0$, one can build a $\mathcal{D}_{\approx Near} = \mathcal{D}_{\approx Near}(\mathsf{P}, r, (1+\varepsilon)r)$, such that given a query point q, one can decide:*

(A) If $\mathbf{b}(q, r) \cap \mathsf{P} \neq \emptyset$, then $\mathcal{D}_{\approx Near}$ returns a point $u \in \mathsf{P}$, such that $\mathsf{d}_H(u, q) \leq (1+\varepsilon)r$.

(B) If $\mathbf{b}(q, (1+\varepsilon)r) \cap \mathsf{P} = \emptyset$, then $\mathcal{D}_{\approx Near}$ returns the result that no point is within distance $\leq r$ from q.

In any other case, any of the answers is correct. The query time is $O(dn^{1/(1+\varepsilon)} \log n)$ and the space used is $O\left(dn + n^{1+1/(1+\varepsilon)} \log n\right)$. The result returned is correct with high probability.

18.3.3. ANN in high-dimensional euclidean space

Unlike the binary hypercube case, where we could just do direct binary search on the distances, here we need to use the reduction from ANN to near neighbor queries.

18.3.3.1. The result

Plugging the above into known reduction from approximate nearest-neighbor to near-neighbor queries, yields the following:

Corollary 18.3.5. *Given a set P of n points in $\mathbb{R}^d$, one can construct a data-structure $\mathcal{D}$ that answers $(1+\varepsilon)$-ANN queries, by performing $O(\log(n/\varepsilon))$ $(1+\varepsilon)$-approximate near neighbor queries. The total number of points stored at these approximate near neighbor data-structure of $\mathcal{D}$ is $O(n\varepsilon^{-1}\log(n/\varepsilon))$.*

This in turn leads to the following:

Theorem 18.3.6. *Given a set P of n points in $\mathbb{R}^d$ and parameters $\varepsilon > 0$ and $r > 0$, one can build an ANN data-structure using*

$$O\left(dn + n^{1+1/(1+\varepsilon)}\varepsilon^{-2}\log^3(n/\varepsilon)\right)$$

space, such that given a query point q, one can returns a $(1+\varepsilon)$-ANN in P in

$$O\left(dn^{1/(1+\varepsilon)}(\log n)\log\frac{n}{\varepsilon}\right)$$

time. The result returned is correct with high probability.

The construction time is $O\left(dn^{1+1/(1+\varepsilon)}\varepsilon^{-2}\log^3(n/\varepsilon)\right)$.

18.4. Bibliographical notes

Section 18.1 follows the exposition of Indyk and Motwani [IM98]. Kushilevitz *et al.* [KOR00] offered an alternative data-structure with somewhat inferior performance. It is quite surprising that one can perform approximate nearest neighbor queries in high dimensions in time and space polynomial in the dimension (which is sublinear in the number of points). One can reduce the approximate near neighbor in Euclidean space to the same question on the hypercube "directly" (we show the details below). However, doing the LSH directly on the Euclidean space is more efficient.

The value of the results shown in this chapter depends to a large extent on the reader's perspective. Indeed, for a small value of $\varepsilon > 0$, the query time $O(dn^{1/(1+\varepsilon)})$ is very close to linear dependency on n and is almost equivalent to just scanning the points. Thus, from the low-dimensional perspective, where ε is assumed to be small, this result is slightly sublinear. On the other hand, if one is willing to pick ε to be large (say 10), then the result is clearly better than the naive algorithm, suggesting running time for an ANN query which takes (roughly) $O\left(n^{1/11}\right)$ time.

The idea of doing locality sensitive hashing directly on the Euclidean space, as done in Section 18.3, is not shocking after seeing the Johnson-Lindenstrauss lemma. Our description follows the paper of Datar *et al.* [DIIM04]. In particular, the current analysis which relies on computerized estimates is far from being satisfactory. It would be nice to have a simpler and more elegant scheme for this case. This is an open problem for further research.

Currently, the best LSH construction in $\mathbb{R}^d$ is due to Andoni and Indyk [AI06]. Its space usage is bounded by $O\left(dn + n^{1+1/(1+\varepsilon)^2+o(1)}\right)$ and its query time is bounded by $O\left(dn^{1/(1+\varepsilon)^2+o(1)}\right)$. This (almost) matches the lower bound of Motwani *et al.* [MNP06]. For a nice survey on LSH see [AI08].

From approximate near neighbor in $\mathbb{R}^d$ to approximate near neighbor on the hypercube.

The reduction is quite involved, and we only sketch the details. Let P be a set of n points in $\mathbb{R}^d$. We first reduce the dimension to $k = O(\varepsilon^{-2} \log n)$ using the Johnson-Lindenstrauss lemma. Next, we embed this space into $\ell_1^{k'}$ (this is the space $\mathbb{R}^k$, where distances are the L_1 metric instead of the regular L_2 metric), where $k' = O(k/\varepsilon^2)$. This can be done with distortion $(1+\varepsilon)$.

Let Q' be the resulting set of points in $\mathbb{R}^{k'}$. We want to solve approximate near neighbor queries on this set of points, for radius r. As a first step, we partition the space into cells by taking a grid with sidelength (say) $k'r$ and randomly translating it, clipping the points inside each grid cell. It is now sufficient to solve the approximate near neighbor problem inside this grid cell (which has bounded diameter as a function of r), since with small probability the result would be correct. We amplify the probability by repeating this a polylogarithmic number of times.

Thus, we can assume that P is contained inside a cube of sidelength $\leq k'nr$, it is in $\mathbb{R}^{k'}$, and the distance metric is the L_1 metric. We next snap the points of P to a grid of sidelength (say) $\varepsilon r/k'$. Thus, every point of P now has an integer coordinate, which is bounded by a polynomial in $\log n$ and $1/\varepsilon$. Next, we write the coordinates of the points of P using unary notation. (Thus, a point $(2, 5)$ would be written as $(00011, 11111)$ assuming the number of bits for each coordinates is 5.) It is now easy to verify that the Hamming distance on the resulting strings is equivalent to the L_1 distance between the points.

Thus, we can solve the near neighbor problem for points in $\mathbb{R}^d$ by solving it on the hypercube under the Hamming distance.

See Indyk and Motwani [IM98] for more details.

We have only scratched the surface of proximity problems in high dimensions. The interested reader is referred to the survey by Indyk [Aga04] for more information.

Chapter 19

Multiplicative Weight Update: Expert Selection

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
December 10, 2019

> Possession of anything new or expensive only reflected a person's lack of theology and geometry; it could even cast doubts upon one's soul.
>
> A confederacy of Dunces, John Kennedy Toole

19.1. The problem: Expert selection

We are given N experts $[\![N]\!] = \{1, 2, \ldots, N\}$. At each time t, an expert i makes a prediction what is going to happen at this time slot. To make things simple, assume the prediction is one of two values, say, 0 or 1. You are going to play this game for a while – at each iteration you are going to get the advice of the N experts, and you are going to select either decision as your own prediction. The purpose here is to come up with a strategy that minimizes the overall number of wrong predictions made.

If there is an expert that is never wrong. This situation is easy – initially start with all n experts as being viable – to this end, we assign $W(i) \leftarrow 1$, for all i. If an expert prediction turns out to be wrong, we set its weight to zero (i.e., it is no longer active). Clearly, if you follow the majority vote of the still viable experts, then at most $\log_2 n$ mistakes would be made, before one isolates the infallible experts.

19.2. Majority vote

The algorithm. Unfortunately, we are unlikely to be in the above scenario – experts makes mistakes. Throwing a way an expert because of a single mistake is a sure way to have no expert remaining. Instead, we are going to moderate our strategy. If expert i is wrong, in a round, we are going to decrease its weight – to be precise, we set $W(i) \leftarrow (1 - \varepsilon)W(i)$, where ε is some parameter. Note, that this weight update is done every round, independent on the decision output in the round. It is now natural, in each round, to compute the total weight of the experts predicting 0, and the total weight of the experts predicting 1, and return the prediction that has a heavier total weight supporting it.

Intuition. The algorithm keeps track of the quality of the experts. The useless experts would have weights very close to zero.

Analysis. We need the following easy calculation.

Lemma 19.2.1. *For $x \in [0, 1/2]$, we have $1 - x \geq \exp(-x - x^2)$.*

Proof: For $x \in (-1, 1)$, the Taylor expansion of $\ln(1 + x)$ is $\sum_{i=1}^{\infty} (-1)^{i+1} \frac{x^i}{i}$. As such, for $x \in [0, 1/2]$ we have

$$\ln(1 - x) = -\sum_{i=1}^{\infty} \frac{x^i}{i} = -x - \frac{x^2}{2} - \frac{x^3}{3} \cdots \geq -x - x^2,$$

since $x^{2+i}/(2 + i) \leq x^2/2^i \iff x^i/(2 + i) \leq 1/2^i$, which is obviously true as $x \leq 1/2$. ∎

Lemma 19.2.2. *Let assume we have N experts. Let $\boldsymbol{\beta}_t$ be the number of the mistakes the algorithm performs, and let $\beta_t(i)$ be the number of mistakes made by the ith expert, for $i \in [\![n]\!]$ (both till time t). Then, if we run this algorithm for T rounds, we have*

$$\forall i \in [\![n]\!] \qquad \boldsymbol{\beta}_T \leq 2(1 + \varepsilon)\beta_T(i) + \frac{2 \log N}{\varepsilon}.$$

Proof: Let Φ_t be the total weight of the experts at the beginning of round t. Observe that $\Phi_1 = N$, and if a mistake was made in the t round, then

$$\Phi_{t+1} \leq (1 - \varepsilon/2)\Phi_t \leq \exp(-\varepsilon \boldsymbol{\beta}_{t+1}/2)N.$$

On the other hand, an expert i made $\beta_i(t)$ mistakes in the first t rounds, and as such its weight, at this point in time, is $(1 - \varepsilon)^{\beta_t(i)}$. We thus have, at time T, and for any i, that

$$\exp\Big(-\big(\varepsilon + \varepsilon^2\big)\beta_T(i)\Big) \leq (1 - \varepsilon)^{\beta_T(i)} \leq \Phi_T \leq \exp\left(-\frac{\varepsilon \boldsymbol{\beta}_T}{2}\right)N.$$

Taking ln of both sides, we have $-\big(\varepsilon + \varepsilon^2\big)\beta_T(i) \leq -\frac{\varepsilon \boldsymbol{\beta}_T}{2} + \ln N. \iff \boldsymbol{\beta}_T \leq 2(1 + \varepsilon)\beta_T(i) + 2\frac{\ln N}{\varepsilon}$. ∎

19.3. Randomized weighted majority

Let $W_t(i)$ be the weight assigned to the ith expert with in the beginning of the tth round. We modify the algorithm to choose expert i, at round t, with probability $W_t(i)/\Phi_t$. That is, the algorithm randomly choose an expert to follow according to their weights. Unlike before, all the experts that are wrong in a round get a weight decrease.

Proof: We have that $\Phi_t = \sum_{i=1}^{N} W_t(i)$. Let $m_t(i) = 1$ be a an indicator variable that is one if and only if expert i made a mistake at round t. Similarly, let $\mathbf{m}_t = 1 \iff$ the algorithm made a mistake at round t. By definition, we have that

$$\mathbb{E}[\mathbf{m}_t] = \sum_{i=1}^{N} \mathbb{P}[i \text{ expert chosen}]\, m_t(i) = \sum_{i=1}^{N} \frac{W_t(i)}{\Phi_t}\, m_t(i).$$

We then have that

$$W_{t+1}(i) = \big(1 - \varepsilon m_t(i)\big)W_t(i).$$

As such, we have $\Phi_{t+1} = \sum_{i=1}^{N} W_{t+1}(i)$, and

$$\Phi_{t+1} = \sum_{i=1}^{N} \big(1 - \varepsilon m_t(i)\big) W_t(i) = \Phi_t - \varepsilon \sum_{i=1}^{N} m_t(i) W_t(i) = \Phi_t - \varepsilon \Phi_t \sum_{i=1}^{N} m_t(i) \frac{W_t(i)}{\Phi_t} = \Big(1 - \varepsilon \, \mathbb{E}[\mathbf{m}_t]\Big) \, \Phi_t.$$

We now follow the same argument as before

$$(1-\varepsilon)^{\beta_T(i)} \le \Phi_T \le N \prod_{t=1}^{T} \Big(1 - \varepsilon \, \mathbb{E}[\mathbf{m}_t]\Big) \le N \exp(-\varepsilon \, \mathbb{E}[\boldsymbol{\beta}_T]) \implies (-\varepsilon - \varepsilon^2)\beta_T(i) \le \ln N - \varepsilon \, \mathbb{E}[\boldsymbol{\beta}_T]$$

$$\implies \mathbb{E}[\boldsymbol{\beta}_T] \le (1+\varepsilon)\beta_T(i) + \frac{\ln N}{\varepsilon}.$$ ■

19.4. Bibliographical notes

Chapter 20

On Complexity, Sampling, and ε-Nets and ε-Samples

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
December 10, 2019

> "I've never touched the hard stuff, only smoked grass a few times with the boys to be polite, and that's all, though ten is the age when the big guys come around teaching you all sorts to things. But happiness doesn't mean much to me, I still think life is better. Happiness is a mean son of a bitch and needs to be put in his place. Him and me aren't on the same team, and I'm cutting him dead. I've never gone in for politics, because somebody always stand to gain by it, but happiness is an even crummier racket, and their ought to be laws to put it out of business."
>
> Momo, Emile Ajar

In this chapter we will try to quantify the notion of geometric complexity. It is intuitively clear that a ● (i.e., disk) is a simpler shape than an ⬬ (i.e., ellipse), which is in turn simpler than a ☻ (i.e., smiley). This becomes even more important when we consider several such shapes and how they interact with each other. As these examples might demonstrate, this notion of complexity is somewhat elusive.

To this end, we show that one can capture the structure of a distribution/point set by a small subset. The size here would depend on the complexity of the shapes/ranges we care about, but surprisingly it would be independent of the size of the point set.

20.1. VC dimension

Definition 20.1.1. A ***range space*** S is a pair $(\mathsf{X}, \mathcal{R})$, where X is a ***ground set*** (finite or infinite) and $\mathcal{R}$ is a (finite or infinite) family of subsets of X. The elements of X are ***points*** and the elements of $\mathcal{R}$ are ***ranges***.

Our interest is in the size/weight of the ranges in the range space. For technical reasons, it will be easier to consider a finite subset x as the underlining ground set.

Definition 20.1.2. Let $\mathsf{S} = (\mathsf{X}, \mathcal{R})$ be a range space, and let x be a finite (fixed) subset of X. For a range $\mathbf{r} \in \mathcal{R}$, its ***measure*** is the quantity

$$\overline{m}(\mathbf{r}) = \frac{|\mathbf{r} \cap \mathsf{x}|}{|\mathsf{x}|}.$$

While x is finite, it might be very large. As such, we are interested in getting a good estimate to $\overline{m}(\mathbf{r})$ by using a more compact set to represent the range space.

Definition 20.1.3. Let $\mathsf{S} = (\mathsf{X}, \mathcal{R})$ be a range space. For a subset N (which might be a multi-set) of x, its ***estimate*** of the measure of $\overline{m}(\mathbf{r})$, for $\mathbf{r} \in \mathcal{R}$, is the quantity

$$\overline{s}(\mathbf{r}) = \frac{|\mathbf{r} \cap N|}{|N|}.$$

The main purpose of this chapter is to come up with methods to generate a sample N, such that $\overline{m}(\mathbf{r}) \approx \overline{s}(\mathbf{r})$, for all the ranges $\mathbf{r} \in \mathcal{R}$.

It is easy to see that in the worst case, no sample can capture the measure of all ranges. Indeed, given a sample N, consider the range $\mathsf{x} \setminus N$ that is being completely missed by N. As such, we need to concentrate on range spaces that are "low dimensional", where not all subsets are allowable ranges. The notion of VC dimension (named after Vapnik and Chervonenkis [VC71]) is one way to limit the complexity of a range space.

Definition 20.1.4. Let $\mathsf{S} = (\mathsf{X}, \mathcal{R})$ be a range space. For $Y \subseteq \mathsf{X}$, let

$$\mathcal{R}_{|Y} = \left\{ \mathbf{r} \cap Y \;\middle|\; \mathbf{r} \in \mathcal{R} \right\} \tag{20.1}$$

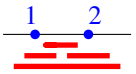
denote the ***projection*** of $\mathcal{R}$ on Y. The range space S projected to Y is $\mathsf{S}_{|Y} = (Y, \mathcal{R}_{|Y})$.

If $\mathcal{R}_{|Y}$ contains all subsets of Y (i.e., if Y is finite, we have $\left|\mathcal{R}_{|Y}\right| = 2^{|Y|}$), then Y is ***shattered*** by $\mathcal{R}$ (or equivalently Y is shattered by S).

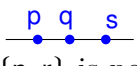
The ***Vapnik-Chervonenkis*** dimension (or VC ***dimension***) of S, denoted by $\dim_{\mathsf{VC}}(\mathsf{S})$, is the maximum cardinality of a shattered subset of X. If there are arbitrarily large shattered subsets, then $\dim_{\mathsf{VC}}(S) = \infty$.

20.1.1. Examples

Intervals. Consider the set X to be the real line, and consider $\mathcal{R}$ to be the set of all intervals on the real line. Consider the set $Y = \{1, 2\}$. Clearly, one can find four intervals that contain all possible subsets of Y. Formally, the projection $\mathcal{R}_{|Y} = \{\{\}, \{1\}, \{2\}, \{1, 2\}\}$. The intervals realizing each of these subsets are depicted on the right.

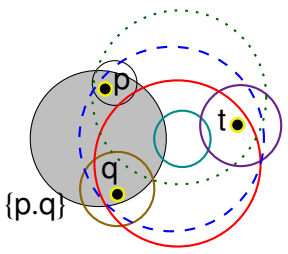


However, this is false for a set of three points $B = \{\mathsf{p}, \mathsf{q}, \mathsf{r}\}$, since there is no interval that can contain the two extreme points p and r without also containing q. Namely, the subset $\{\mathsf{p}, \mathsf{r}\}$ is not realizable for intervals, implying that the largest shattered set by the range space (real line, intervals) is of size two. We conclude that the VC dimension of this space is two.

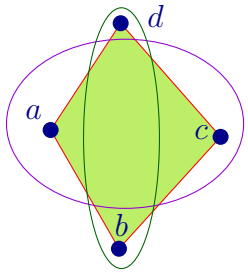


Disks. Let $\mathsf{X} = \mathbb{R}^2$, and let $\mathcal{R}$ be the set of disks in the plane. Clearly, for any three points in the plane (in general position), denoted by p, q, and r, one can find eight disks that realize all possible 2^3 different subsets. See the figure on the right.

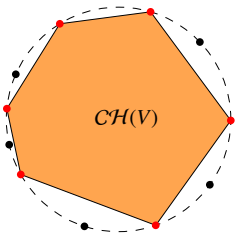
But can disks shatter a set with four points? Consider such a set P of four points. If the convex hull of P has only three points on its boundary, then the subset X having only those three vertices (i.e., it does not include the middle point) is impossible, by convexity. Namely, there is no disk that contains only the points of X without the middle point.



Alternatively, if all four points are vertices of the convex hull and they are a, b, c, d along the boundary of the convex hull, either the set $\{a, c\}$ or the set $\{b, d\}$ is not realizable. Indeed, if both options are realizable, then consider the two disks D_1 and D_2 that realize those assignments. Clearly, ∂D_1 and ∂D_2 must intersect in four points, but this is not possible, since two circles have at most two intersection points. See the figure on the left. Hence the VC dimension of this range space is 3.



Convex sets. Consider the range space $\mathsf{S} = (\mathbb{R}^2, \mathcal{R})$, where $\mathcal{R}$ is the set of all (closed) convex sets in the plane. We claim that $\dim_{\mathsf{VC}}(\mathsf{S}) = \infty$. Indeed, consider a set U of n points $p_1, \ldots, p_n$ all lying on the boundary of the unit circle in the plane. Let V be any subset of U, and consider the convex hull $\mathcal{CH}(V)$. Clearly, $\mathcal{CH}(V) \in \mathcal{R}$, and furthermore, $\mathcal{CH}(V) \cap U = V$. Namely, any subset of U is realizable by S. Thus, S can shatter sets of arbitrary size, and its VC dimension is unbounded.



Complement. Consider the range space $\mathsf{S} = (\mathsf{X}, \mathcal{R})$ with $\delta = \dim_{\mathsf{VC}}(\mathsf{S})$. Next, consider the complement space, $\overline{\mathsf{S}} = (\mathsf{X}, \overline{\mathcal{R}})$, where

$$\overline{\mathcal{R}} = \left\{ \mathsf{X} \setminus \mathbf{r} \;\middle|\; \mathbf{r} \in \mathcal{R} \right\}.$$

Namely, the ranges of $\overline{\mathsf{S}}$ are the complement of the ranges in S. What is the VC dimension of $\overline{\mathsf{S}}$? Well, a set $B \subseteq \mathsf{X}$ is shattered by $\overline{\mathsf{S}}$ if and only if it is shattered by S. Indeed, if S shatters B, then for any $Z \subseteq B$, we have that $(B \setminus Z) \in \mathcal{R}_{|B}$, which implies that $Z = B \setminus (B \setminus Z) \in \overline{\mathcal{R}}_{|B}$. Namely, $\overline{\mathcal{R}}_{|B}$ contains all the subsets of B, and $\overline{\mathsf{S}}$ shatters B. Thus, $\dim_{\mathsf{VC}}(\overline{\mathsf{S}}) = \dim_{\mathsf{VC}}(\mathsf{S})$.

Lemma 20.1.5. *For a range space $\mathsf{S} = (X, \mathcal{R})$ we have that $\dim_{\mathsf{VC}}(\mathsf{S}) = \dim_{\mathsf{VC}}\left(\overline{\mathsf{S}}\right)$, where $\overline{\mathsf{S}}$ is the complement range space.*

20.1.1.1. Halfspaces

Let $\mathsf{S} = (\mathsf{X}, \mathcal{R})$, where $\mathsf{X} = \mathbb{R}^d$ and $\mathcal{R}$ is the set of all (closed) halfspaces in $\mathbb{R}^d$. We need the following technical claim.

Claim 20.1.6. *Let $\mathsf{P} = \{\mathsf{p}_1, \ldots, \mathsf{p}_{d+2}\}$ be a set of $d+2$ points in $\mathbb{R}^d$. There are real numbers $\beta_1, \ldots, \beta_{d+2}$, not all of them zero, such that $\sum_i \beta_i \mathsf{p}_i = 0$ and $\sum_i \beta_i = 0$.*

Proof: Indeed, set $\mathsf{q}_i = (\mathsf{p}_i, 1)$, for $i = 1, \ldots, d+2$. Now, the points $\mathsf{q}_1, \ldots, \mathsf{q}_{d+2} \in \mathbb{R}^{d+1}$ are linearly dependent, and there are coefficients $\beta_1, \ldots, \beta_{d+2}$, not all of them zero, such that $\sum_{i=1}^{d+2} \beta_i \mathsf{q}_i = 0$. Considering only the first d coordinates of these points implies that $\sum_{i=1}^{d+2} \beta_i \mathsf{p}_i = 0$. Similarly, by considering only the $(d+1)$st coordinate of these points, we have that $\sum_{i=1}^{d+2} \beta_i = 0$. ∎

To see what the VC dimension of halfspaces in $\mathbb{R}^d$ is, we need the following result of Radon. (For a reminder of the formal definition of convex hulls, see Definition 20.5.1.)

Theorem 20.1.7 (Radon's theorem). *Let $\mathsf{P} = \{\mathsf{p}_1, \ldots, \mathsf{p}_{d+2}\}$ be a set of $d+2$ points in $\mathbb{R}^d$. Then, there exist two disjoint subsets C and D of P, such that $\mathcal{CH}(C) \cap \mathcal{CH}(D) \neq \emptyset$ and $C \cup D = \mathsf{P}$.*

Proof: By Claim 20.1.6 there are real numbers $\beta_1, \ldots, \beta_{d+2}$, not all of them zero, such that $\sum_i \beta_i \mathsf{p}_i = 0$ and $\sum_i \beta_i = 0$.

Assume, for the sake of simplicity of exposition, that $\beta_1, \ldots, \beta_k \geq 0$ and $\beta_{k+1}, \ldots, \beta_{d+2} < 0$. Furthermore, let $\mu = \sum_{i=1}^{k} \beta_i = -\sum_{i=k+1}^{d+2} \beta_i$. We have that

$$\sum_{i=1}^{k} \beta_i \mathsf{p}_i = -\sum_{i=k+1}^{d+2} \beta_i \mathsf{p}_i.$$

In particular, $v = \sum_{i=1}^{k} (\beta_i/\mu)\mathsf{p}_i$ is a point in $\mathcal{CH}(\{\mathsf{p}_1, \ldots, \mathsf{p}_k\})$. Furthermore, for the same point v we have $v = \sum_{i=k+1}^{d+2} -(\beta_i/\mu)\mathsf{p}_i \in \mathcal{CH}(\{\mathsf{p}_{k+1}, \ldots, \mathsf{p}_{d+2}\})$. We conclude that v is in the intersection of the two convex hulls, as required. ■

The following is a trivial observation, and yet we provide a proof to demonstrate it is true.

Lemma 20.1.8. *Let $\mathsf{P} \subseteq \mathbb{R}^d$ be a finite set, let r be any point in $\mathcal{CH}(\mathsf{P})$, and let h^+ be a halfspace of $\mathbb{R}^d$ containing r. Then there exists a point of P contained inside h^+.*

Proof: The halfspace h^+ can be written as $h^+ = \left\{\mathsf{t} \in \mathbb{R}^d \,\middle|\, \langle \mathsf{t}, v\rangle \leq c\right\}$. Now $\mathsf{r} \in \mathcal{CH}(\mathsf{P}) \cap h^+$, and as such there are numbers $\alpha_1, \ldots, \alpha_m \geq 0$ and points $\mathsf{p}_1, \ldots, \mathsf{p}_m \in \mathsf{P}$, such that $\sum_i \alpha_i = 1$ and $\sum_i \alpha_i \mathsf{p}_i = \mathsf{r}$. By the linearity of the dot product, we have that

$$\mathsf{r} \in h^+ \implies \langle \mathsf{r}, v\rangle \leq c \implies \left\langle \sum_{i=1}^{m} \alpha_i \mathsf{p}_i, v \right\rangle \leq c \implies \beta = \sum_{i=1}^{m} \alpha_i \langle \mathsf{p}_i, v\rangle \leq c.$$

Setting $\beta_i = \langle \mathsf{p}_i, v\rangle$, for $i = 1, \ldots, m$, the above implies that β is a weighted average of $\beta_1, \ldots, \beta_m$. In particular, there must be a β_i that is no larger than the average. That is $\beta_i \leq c$. This implies that $\langle \mathsf{p}_i, v\rangle \leq c$. Namely, $\mathsf{p}_i \in h^+$ as claimed. ■

Let S be the range space having $\mathbb{R}^d$ as the ground set and all the close halfspaces as ranges. Radon's theorem implies that if a set Q of $d+2$ points is being shattered by S, then we can partition this set Q into two disjoint sets Y and Z such that $\mathcal{CH}(Y) \cap \mathcal{CH}(Z) \neq \emptyset$. In particular, let r be a point in $\mathcal{CH}(Y) \cap \mathcal{CH}(Z)$. If a halfspace h^+ contains all the points of Y, then $\mathcal{CH}(Y) \subseteq h^+$, since a halfspace is a convex set. Thus, any halfspace h^+ containing all the points of Y will contain the point $\mathsf{r} \in \mathcal{CH}(Y)$. But $\mathsf{r} \in \mathcal{CH}(Z) \cap h^+$, and this implies that a point of Z must lie in h^+, by Lemma 20.1.8. Namely, the subset $Y \subseteq \mathsf{Q}$ cannot be realized by a halfspace, which implies that Q cannot be shattered. Thus $\dim_{\mathsf{VC}}(S) < d+2$. It is also easy to verify that the regular simplex with $d+1$ vertices is shattered by S. Thus, $\dim_{\mathsf{VC}}(S) = d+1$.

20.2. Shattering dimension and the dual shattering dimension

The main property of a range space with bounded VC dimension is that the number of ranges for a set of n elements grows polynomially in n (with the power being the dimension) instead of exponentially. Formally, let the ***growth function*** be

$$\mathcal{G}_\delta(n) = \sum_{i=0}^{\delta} \binom{n}{i} \leq \sum_{i=0}^{\delta} \frac{n^i}{i!} \leq n^\delta, \tag{20.2}$$

for $\delta > 1$ (the cases where $\delta = 0$ or $\delta = 1$ are not interesting and we will just ignore them). Note that for all $n, \delta \geq 1$, we have $\mathcal{G}_\delta(n) = \mathcal{G}_\delta(n-1) + \mathcal{G}_{\delta-1}(n-1)$[①].

[①]Here is a cute (and standard) counting argument: $\mathcal{G}_\delta(n)$ is just the number of different subsets of size at most δ out of n elements. Now, we either decide to not include the first element in these subsets (i.e., $\mathcal{G}_\delta(n-1)$) or, alternatively, we include the first element in these subsets, but then there are only $\delta - 1$ elements left to pick (i.e., $\mathcal{G}_{\delta-1}(n-1)$).

Lemma 20.2.1 (Sauer's lemma). *If $(X, \mathcal{R})$ is a range space of* VC *dimension δ with $|X| = n$, then $|\mathcal{R}| \leq \mathcal{G}_\delta(n)$.*

Proof: The claim trivially holds for $\delta = 0$ or $n = 0$.

Let x be any element of X, and consider the sets

$$\mathcal{R}_x = \left\{ \mathbf{r} \setminus \{x\} \;\middle|\; \mathbf{r} \cup \{x\} \in \mathcal{R} \text{ and } \mathbf{r} \setminus \{x\} \in \mathcal{R} \right\} \quad \text{and} \quad \mathcal{R} \setminus x = \left\{ \mathbf{r} \setminus \{x\} \;\middle|\; \mathbf{r} \in \mathcal{R} \right\}.$$

Observe that $|\mathcal{R}| = |\mathcal{R}_x| + |\mathcal{R} \setminus x|$. Indeed, we charge the elements of $\mathcal{R}$ to their corresponding element in $\mathcal{R} \setminus x$. The only bad case is when there is a range $\mathbf{r}$ such that both $\mathbf{r} \cup \{x\} \in \mathcal{R}$ and $\mathbf{r} \setminus \{x\} \in \mathcal{R}$, because then these two distinct ranges get mapped to the same range in $\mathcal{R} \setminus x$. But such ranges contribute exactly one element to $\mathcal{R}_x$. Similarly, every element of $\mathcal{R}_x$ corresponds to two such "twin" ranges in $\mathcal{R}$.

Observe that $(X \setminus \{x\}, \mathcal{R}_x)$ has VC dimension $\delta - 1$, as the largest set that can be shattered is of size $\delta - 1$. Indeed, any set $B \subset \mathsf{X} \setminus \{x\}$ shattered by $\mathcal{R}_x$ implies that $B \cup \{x\}$ is shattered in $\mathcal{R}$.

Thus, we have

$$|\mathcal{R}| = |\mathcal{R}_x| + |\mathcal{R} \setminus x| \leq \mathcal{G}_{\delta-1}(n-1) + \mathcal{G}_\delta(n-1) = \mathcal{G}_\delta(n),$$

by induction. ∎

Interestingly, Lemma 20.2.1 is tight.

Next, we show pretty tight bounds on $\mathcal{G}_\delta(n)$. The proof is technical and not very interesting, and it is delegated to Section 20.4.

Lemma 20.2.2. *For $n \geq 2\delta$ and $\delta \geq 1$, we have* $\left(\frac{n}{\delta}\right)^\delta \leq \mathcal{G}_\delta(n) \leq 2\left(\frac{ne}{\delta}\right)^\delta$, *where* $\mathcal{G}_\delta(n) = \sum_{i=0}^{\delta} \binom{n}{i}$.

Definition 20.2.3 (Shatter function). Given a range space $\mathsf{S} = (\mathsf{X}, \mathcal{R})$, its ***shatter function*** $\pi_{\mathsf{S}}(m)$ is the maximum number of sets that might be created by S when restricted to subsets of size m. Formally,

$$\pi_{\mathsf{S}}(m) = \max_{\substack{B \subset \mathsf{X} \\ |B| = m}} \left| \mathcal{R}_{|B} \right|;$$

see Eq. (20.1).

Our arch-nemesis in the following is the function $x/\ln x$. The following lemma states some properties of this function, and its proof is left as an exercise.

Lemma 20.2.4. *For the function $f(x) = x/\ln x$ the following hold.*

(A) $f(x)$ is monotonically increasing for $x \geq e$.
(B) $f(x) \geq e$, for $x > 1$.
(C) For $u \geq \sqrt{e}$, if $f(x) \leq u$, then $x \leq 2u \ln u$.
(D) For $u \geq \sqrt{e}$, if $x > 2u \ln u$, then $f(x) > u$.
(E) For $u \geq e$, if $f(x) \geq u$, then $x \geq u \ln u$.

20.2.1. Mixing range spaces

Lemma 20.2.5. *Let $\mathsf{S} = (X, \mathcal{R})$ and $\mathsf{T} = (X, \mathcal{R}')$ be two range spaces of* VC *dimension δ and δ', respectively, where $\delta, \delta' > 1$. Let $\widehat{\mathcal{R}} = \{\mathbf{r} \cup \mathbf{r}' \mid \mathbf{r} \in \mathcal{R}, \mathbf{r}' \in \mathcal{R}'\}$. Then, for the range space $\widehat{\mathsf{S}} = (X, \widehat{\mathcal{R}})$, we have that $\dim_{\mathsf{VC}}(\widehat{\mathsf{S}}) = O(\delta + \delta')$.*

Proof: As a warm-up exercise, we prove a somewhat weaker bound here of $O((\delta+\delta')\log(\delta+\delta'))$. The stronger bound follows from Theorem 20.2.6 below. Let B be a set of n points in X that are shattered by $\widehat{\mathsf{S}}$. There are at most $\mathcal{G}_\delta(n)$ and $\mathcal{G}_{\delta'}(n)$ different ranges of B in the range sets $\mathcal{R}_{|B}$ and $\mathcal{R}'_{|B}$, respectively, by Lemma 20.2.1. Every subset C of B realized by $\widehat{r} \in \widehat{\mathcal{R}}$ is a union of two subsets $B \cap \mathbf{r}$ and $B \cap \mathbf{r}'$, where $\mathbf{r} \in R$ and $\mathbf{r}' \in \mathcal{R}'$, respectively. Thus, the number of different subsets of B realized by $\widehat{\mathsf{S}}$ is bounded by $\mathcal{G}_\delta(n)\mathcal{G}_{\delta'}(n)$. Thus, $2^n \le n^\delta n^{\delta'}$, for $\delta, \delta' > 1$. We conclude that $n \le (\delta+\delta')\lg n$, which implies that $n = O((\delta+\delta')\log(\delta+\delta'))$, by Lemma 20.2.4(C). ∎

Interestingly, one can prove a considerably more general result with tighter bounds. The required computations are somewhat more painful.

Theorem 20.2.6. *Let $\mathsf{S}_1 = (X, \mathcal{R}^1), \ldots, \mathsf{S}_k = (X, \mathcal{R}^k)$ be range spaces with VC dimension $\delta_1, \ldots, \delta_k$, respectively. Next, let $f(\mathbf{r}_1, \ldots, \mathbf{r}_k)$ be a function that maps any k-tuple of sets $\mathbf{r}_1 \in \mathcal{R}^1, \ldots, \mathbf{r}_k \in \mathcal{R}^k$ into a subset of X. Here, the function f is restricted to be defined by a sequence of set operations like complement, intersection and union. Consider the range set*

$$\mathcal{R}' = \left\{ f(\mathbf{r}_1, \ldots, \mathbf{r}_k) \;\middle|\; \mathbf{r}_1 \in \mathcal{R}_1, \ldots, \mathbf{r}_k \in \mathcal{R}_k \right\}$$

and the associated range space $\mathsf{T} = (X, \mathcal{R}')$. Then, the VC dimension of T is bounded by $O(k\delta \lg k)$, where $\delta = \max_i \delta_i$.

Proof: Assume a set $Y \subseteq \mathsf{X}$ of size t is being shattered by $\mathcal{R}'$, and observe that

$$\begin{aligned}\left|\mathcal{R}'_{|Y}\right| &\le \left| \left\{ (\mathbf{r}_1, \ldots, \mathbf{r}_k) \;\middle|\; \mathbf{r}_1 \in \mathcal{R}^1_{|Y}, \ldots, \mathbf{r}_k \in \mathcal{R}^k_{|Y} \right\} \right| \le \left|\mathcal{R}^1_{|Y}\right| \cdots \left|\mathcal{R}^k_{|Y}\right| \le \mathcal{G}_{\delta_1}(t) \cdot \mathcal{G}_{\delta_2}(t) \cdots \mathcal{G}_{\delta_k}(t) \\ &\le \left(\mathcal{G}_\delta(t)\right)^k \le \left(2\left(te/\delta\right)^\delta\right)^k,\end{aligned}$$

by Lemma 20.2.1 and Lemma 20.2.2. On the other hand, since Y is being shattered by $\mathcal{R}'$, this implies that $\left|\mathcal{R}'_{|Y}\right| = 2^t$. Thus, we have the inequality $2^t \le \left(2(te/\delta)^\delta\right)^k$, which implies $t \le k(1+\delta\lg(te/\delta))$. Assume that $t \ge e$ and $\delta\lg(te/\delta) \ge 1$ since otherwise the claim is trivial, and observe that $t \le k(1+\delta\lg(te/\delta)) \le 3k\delta\lg(t/\delta)$. Setting $x = t/\delta$, we have

$$\frac{t}{\delta} \le 3k\frac{\ln(t/\delta)}{\ln 2} \le 6k\ln\frac{t}{\delta} \implies \frac{x}{\ln x} \le 6k \implies x \le 2\cdot 6k\ln(6k) \implies x \le 12k\ln(6k),$$

by Lemma 20.2.4(C). We conclude that $t \le 12\delta k\ln(6k)$, as claimed. ∎

Corollary 20.2.7. *Let $\mathsf{S} = (X, \mathcal{R})$ and $\mathsf{T} = (X, \mathcal{R}')$ be two range spaces of VC dimension δ and δ', respectively, where $\delta, \delta' > 1$. Let $\widehat{\mathcal{R}} = \left\{ \mathbf{r} \cap \mathbf{r}' \;\middle|\; \mathbf{r} \in \mathcal{R}, \mathbf{r}' \in \mathcal{R}' \right\}$. Then, for the range space $\widehat{\mathsf{S}} = (X, \widehat{\mathcal{R}})$, we have that $\dim_{\mathsf{VC}}(\widehat{\mathsf{S}}) = O(\delta+\delta')$.*

Corollary 20.2.8. *Any finite sequence of combining range spaces with finite VC dimension (by intersecting, complementing, or taking their union) results in a range space with a finite VC dimension.*

20.3. On ε-nets and ε-sampling

20.3.1. ε-nets and ε-samples

Definition 20.3.1 (ε-sample). Let $\mathsf{S} = (\mathsf{X}, \mathcal{R})$ be a range space, and let x be a finite subset of X. For $0 \leq \varepsilon \leq 1$, a subset $C \subseteq \mathsf{x}$ is an ***ε-sample*** for x if for any range $\mathbf{r} \in \mathcal{R}$, we have

$$\left| \overline{m}(\mathbf{r}) - \overline{s}(\mathbf{r}) \right| \leq \varepsilon,$$

where $\overline{m}(\mathbf{r}) = |\mathsf{x} \cap \mathbf{r}| / |\mathsf{x}|$ is the measure of $\mathbf{r}$ (see Definition 20.1.2) and $\overline{s}(\mathbf{r}) = |C \cap \mathbf{r}| / |C|$ is the estimate of $\mathbf{r}$ (see Definition 20.1.3). (Here C might be a multi-set, and as such $|C \cap \mathbf{r}|$ is counted with multiplicity.)

As such, an ε-sample is a subset of the ground set x that "captures" the range space up to an error of ε. Specifically, to estimate the fraction of the ground set covered by a range $\mathbf{r}$, it is sufficient to count the points of C that fall inside $\mathbf{r}$.

If X is a finite set, we will abuse notation slightly and refer to C as an ***ε-sample*** for S.

To see the usage of such a sample, consider $\mathsf{x} = \mathsf{X}$ to be, say, the population of a country (i.e., an element of X is a citizen). A range in $\mathcal{R}$ is the set of all people in the country that answer yes to a question (i.e., would you vote for party Y?, would you buy a bridge from me?, questions like that). An ε-sample of this range space enables us to estimate reliably (up to an error of ε) the answers for all these questions, by just asking the people in the sample.

The natural question of course is how to find such a subset of small (or minimal) size.

Theorem 20.3.2 (ε-sample theorem, [VC71]). *There is a positive constant c such that if $(X, \mathcal{R})$ is any range space with VC dimension at most δ, $\mathsf{x} \subseteq X$ is a finite subset and $\varepsilon, \varphi > 0$, then a random subset $C \subseteq \mathsf{x}$ of cardinality*

$$s = \frac{c}{\varepsilon^2}\left(\delta \log \frac{\delta}{\varepsilon} + \log \frac{1}{\varphi}\right)$$

is an ε-sample for x with probability at least $1 - \varphi$.

(In the above theorem, if $s > |\mathsf{x}|$, then we can just take all of x to be the ε-sample.)

Sometimes it is sufficient to have (hopefully smaller) samples with a weaker property – if a range is "heavy", then there is an element in our sample that is in this range.

Definition 20.3.3 (ε-net). A set $N \subseteq \mathsf{x}$ is an ***ε-net*** for x if for any range $\mathbf{r} \in \mathcal{R}$, if $\overline{m}(\mathbf{r}) \geq \varepsilon$ (i.e., $|\mathbf{r} \cap \mathsf{x}| \geq \varepsilon |\mathsf{x}|$), then $\mathbf{r}$ contains at least one point of N (i.e., $\mathbf{r} \cap N \neq \emptyset$).

Theorem 20.3.4 (ε-net theorem, [HW87]). *Let $(X, \mathcal{R})$ be a range space of VC dimension δ, let x be a finite subset of X, and suppose that $0 < \varepsilon \leq 1$ and $\varphi < 1$. Let N be a set obtained by m random independent draws from x, where*

$$m \geq \max\left(\frac{4}{\varepsilon} \lg \frac{4}{\varphi}, \frac{8\delta}{\varepsilon} \lg \frac{16}{\varepsilon} \right). \tag{20.3}$$

Then N is an ε-net for x with probability at least $1 - \varphi$.

(We remind the reader that $\lg = \log_2$.)

The proofs of the above theorems are somewhat involved and we first turn our attention to some applications before presenting the proofs.

Remark 20.3.5. The above two theorems also hold for spaces with shattering dimension at most δ, in which case the sample size is slightly larger. Specifically, for Theorem 20.3.4, the sample size needed is $O\left(\frac{1}{\varepsilon}\lg\frac{1}{\varphi} + \frac{\delta}{\varepsilon}\lg\frac{\delta}{\varepsilon}\right)$.

Remark 20.3.6. The ε-net theorem is a relatively easy consequence (up to constants) of the ε-sample theorem – see bibliographical notes for details.

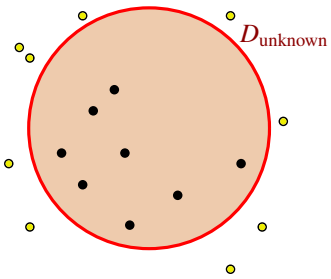
20.3.2. Some applications

We mention two (easy) applications of these theorems, which (hopefully) demonstrate their power.

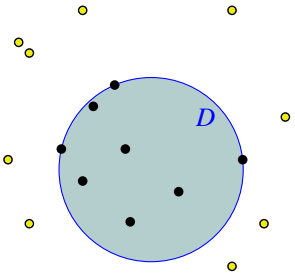
20.3.2.1. Range searching

So, consider a (very large) set of points P in the plane. We would like to be able to quickly decide how many points are included inside a query rectangle. Let us assume that we allow ourselves 1% error. What Theorem 20.3.2 tells us is that there is a subset of *constant size* (that depends only on ε) that can be used to perform this estimation, and it works for *all* query rectangles (we used here the fact that rectangles in the plane have finite VC dimension). In fact, a random sample of this size works with constant probability.

20.3.2.2. Learning a concept

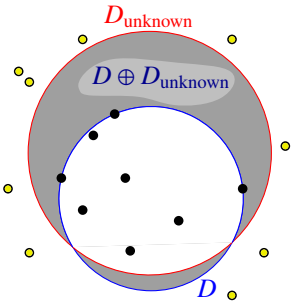


Assume that we have a function f defined in the plane that returns '1' inside an (unknown) disk D_{unknown} and '0' outside it. There is some distribution $\mathcal{D}$ defined over the plane, and we pick points from this distribution. Furthermore, we can compute the function for these labels (i.e., we can compute f for certain values, but it is expensive). For a mystery value $\varepsilon > 0$, to be explained shortly, Theorem 20.3.4 tells us to pick (roughly) $O((1/\varepsilon)\log(1/\varepsilon))$ random points in a sample R from this distribution and to compute the labels for the samples. This is demonstrated in the figure on the right, where black dots are the sample points for which $f(\cdot)$ returned 1.



So, now we have positive examples and negative examples. We would like to find a hypothesis that agrees with all the samples we have and that hopefully is close to the true unknown disk underlying the function f. To this end, compute the smallest disk D that contains the sample labeled by '1' and does not contain any of the '0' points, and let $g : \mathbb{R}^2 \to \{0, 1\}$ be the function g that returns '1' inside the disk and '0' otherwise. We claim that g classifies correctly all but an ε-fraction of the points (i.e., the probability of misclassifying a point picked according to the given distribution is smaller than ε); that is, $\mathbf{Pr}_{\mathsf{p}\in\mathcal{D}}[f(\mathsf{p}) \neq g(\mathsf{p})] \leq \varepsilon$.

Geometrically, the region where g and f disagree is all the points in the symmetric difference between the two disks. That is, $\mathcal{E} = D \oplus D_{\text{unknown}}$; see the figure on the right.



Thus, consider the range space S having the plane as the ground set and the symmetric difference between any two disks as its ranges. By Corollary 20.2.8, this range space has finite VC dimension. Now, consider the (unknown) disk D' that induces f and the region $\mathbf{r} = D_{\text{unknown}} \oplus D$. Clearly, the learned classifier g returns incorrect answers only for points picked inside $\mathbf{r}$.

Thus, the probability of a mistake in the classification is the measure of $\mathbf{r}$ under the distribution $\mathcal{D}$. So, if $\mathbb{P}_{\mathcal{D}}[\mathbf{r}] > \varepsilon$ (i.e., the probability that a sample point falls inside $\mathbf{r}$), then by the ε-net theorem (i.e., Theorem 20.3.4) the set R is an ε-net for S (ignore for the time being the possibility that the random sample fails to be an ε-net) and as such, R contains a point q inside $\mathbf{r}$. But, it is not possible for g (which classifies correctly all the sampled points of R) to make a mistake on q, a contradiction, because by construction, the range $\mathbf{r}$ is where g misclassifies points. We conclude that $\mathbb{P}_{\mathcal{D}}[\mathbf{r}] \leq \varepsilon$, as desired.

Little lies. The careful reader might be tearing his or her hair out because of the above description. First, Theorem 20.3.4 might fail, and the above conclusion might not hold. This is of course true, and in real applications one might use a much larger sample to guarantee that the probability of failure is so small that it can be practically ignored. A more serious issue is that Theorem 20.3.4 is defined only for finite sets. Nowhere does it speak about a continuous distribution. Intuitively, one can approximate a continuous distribution to an arbitrary precision using a huge sample and apply the theorem to this sample as our ground set. A formal proof is more tedious and requires extending the proof of Theorem 20.3.4 to continuous distributions. This is straightforward and we will ignore this topic altogether.

20.4. A better bound on the growth function

In this section, we prove Lemma 20.2.2. Since the proof is straightforward but tedious, the reader can safely skip reading this section.

Lemma 20.4.1. *For any positive integer n, the following hold.*

(i) $(1+1/n)^n \leq e$.

(ii) $(1-1/n)^{n-1} \geq e^{-1}$.

(iii) $n! \geq (n/e)^n$.

(iv) *For any $k \leq n$, we have* $\left(\frac{n}{k}\right)^k \leq \binom{n}{k} \leq \left(\frac{ne}{k}\right)^k$.

Proof: (i) Indeed, $1 + 1/n \leq \exp(1/n)$, since $1 + x \leq e^x$, for $x \geq 0$. As such $(1+1/n)^n \leq \exp(n(1/n)) = e$.

(ii) Rewriting the inequality, we have that we need to prove $\left(\frac{n-1}{n}\right)^{n-1} \geq \frac{1}{e}$. This is equivalent to proving $e \geq \left(\frac{n}{n-1}\right)^{n-1} = \left(1 + \frac{1}{n-1}\right)^{n-1}$, which is our friend from (i).

(iii) Indeed,

$$\frac{n^n}{n!} \leq \sum_{i=0}^{\infty} \frac{n^i}{i!} = e^n,$$

by the Taylor expansion of $e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!}$. This implies that $(n/e)^n \leq n!$, as required.

(iv) Indeed, for any $k \leq n$, we have $\frac{n}{k} \leq \frac{n-1}{k-1}$, as can be easily verified. As such, $\frac{n}{k} \leq \frac{n-i}{k-i}$, for $1 \leq i \leq k-1$. As such,

$$\left(\frac{n}{k}\right)^k \leq \frac{n}{k} \cdot \frac{n-1}{k-1} \cdots \frac{n-k+1}{1} = \binom{n}{k}.$$

As for the other direction, by (iii), we have $\binom{n}{k} \leq \frac{n^k}{k!} \leq \frac{n^k}{\left(\frac{k}{e}\right)^k} = \left(\frac{ne}{k}\right)^k$. ■

Lemma 20.2.2 restated. *For $n \geq 2\delta$ and $\delta \geq 1$, we have $\left(\frac{n}{\delta}\right)^\delta \leq \mathcal{G}_\delta(n) \leq 2\left(\frac{ne}{\delta}\right)^\delta$, where $\mathcal{G}_\delta(n) = \sum_{i=0}^{\delta} \binom{n}{i}$.*

Proof: Note that by Lemma 20.4.1 (iv), we have $\mathcal{G}_\delta(n) = \sum_{i=0}^{\delta} \binom{n}{i} \leq 1 + \sum_{i=1}^{\delta} \left(\frac{ne}{i}\right)^i$. This series behaves like a geometric series with constant larger than 2, since

$$\left(\frac{ne}{i}\right)^i \Big/ \left(\frac{ne}{i-1}\right)^{i-1} = \frac{ne}{i}\left(\frac{i-1}{i}\right)^{i-1} = \frac{ne}{i}\left(1 - \frac{1}{i}\right)^{i-1} \geq \frac{ne}{i}\frac{1}{e} = \frac{n}{i} \geq \frac{n}{\delta} \geq 2,$$

by Lemma 20.4.1. As such, this series is bounded by twice the largest element in the series, implying the claim. ■

20.5. Some required definitions

Definition 20.5.1 (Convex hull). The ***convex hull*** of a set $\mathsf{R} \subseteq \mathbb{R}^d$ is the set of all convex combinations of points of R; that is,

$$\mathcal{CH}(\mathsf{R}) = \left\{ \sum_{i=0}^{m} \alpha_i \mathsf{r}_i \;\middle|\; \forall i \;\; \mathsf{r}_i \in \mathsf{R}, \alpha_i \geq 0, \text{ and } \sum_{j=1}^{m} \alpha_i = 1 \right\}.$$

Chapter 21

Double sampling

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
December 10, 2019

"What does not work when you apply force, would work when you apply even more force."

, Anonymous

21.1. Double sampling

Double sampling is the idea that two random independent samples should look similar, and should *not* be completely different in the way they intersect a certain set. We use the following sampling model, which makes the computations somewhat easier.

Definition 21.1.1. Let $\mathsf{S} = \{f_1, \ldots, f_n\}$ be a set of objects, where the ith object has weight $\omega_i > 0$, for all i. Let $W = \sum_i \omega_i$. For a target size ρ, a ***ρ-sample*** is a random sample $\mathsf{R} \subseteq \mathsf{S}$, where object f_i is picked independently with probability $\rho\omega_i/W$. To simplify the discussion, we assume that $\rho\omega_i/W < 1$. Handling the more general case is easy if somewhat tedious.

Lemma 21.1.2. *Let R_1 and R_2 be two ρ-samples, and consider the merged sample $\mathsf{R} = \mathsf{R}_1 \cup \mathsf{R}_2$. Let $T \subseteq \mathsf{S}$ be a set of m elements. Then, we have that*

$$\mathbb{P}\Big[T \subseteq \mathsf{R}_1 \,\Big|\, T \subseteq \mathsf{R}\Big] \geq \frac{1}{2^m} \qquad \text{and} \qquad \mathbb{P}\Big[T \subseteq \mathsf{R}_1 \text{ and } T \cap \mathsf{R}_2 = \emptyset \,\Big|\, T \subseteq \mathsf{R}\Big] \leq \frac{1}{2^m}.$$

Proof: Consider an object $f \in T$, and observe that $\mathbb{P}\big[f \in \mathsf{R}_1 \text{ or } f \in \mathsf{R}_2 \mid f \in \mathsf{R}\big] = 1$. As such, by symmetry

$$\mathbb{P}\big[f \in \mathsf{R}_1 \mid f \in \mathsf{R}\big] = \mathbb{P}\big[f \in \mathsf{R}_2 \mid f \in \mathsf{R}\big] \geq 1/2,$$

Now, let $T = \{f_1, \ldots, f_m\}$. Since R_1 and R_2 are independent, and each element is being picked independently, we have that

$$\begin{aligned}\mathbb{P}\big[T \subseteq \mathsf{R}_1 \mid T \subseteq \mathsf{R}\big] &= \mathbb{P}\big[f_1, \ldots, f_m \in \mathsf{R}_1 \mid f_1, \ldots, f_m \in \mathsf{R}\big] = \prod_{i=1}^{b} \mathbb{P}\big[f_i \in \mathsf{R}_1 \mid f_1, \ldots, f_m \in \mathsf{R}\big] \\ &= \prod_{i=1}^{b} \mathbb{P}\big[f_i \in \mathsf{R}_1 \mid f_i \in \mathsf{R}\big] \geq \frac{1}{2^m}.\end{aligned}$$

For the second claim, observe that again, by symmetry, we have that

$$\mathbb{P}\big[f \in \mathsf{R}_1 \text{ and } f \notin \mathsf{R}_2 \;\big|\; f \in \mathsf{R}\big] = \mathbb{P}\big[f \notin \mathsf{R}_1 \text{ and } f \in \mathsf{R}_2 \mid f \in \mathsf{R}\big] \leq 1/2,$$

as the two events are disjoint. Now, the claim follows by arguing as above. ∎

21.1.1. Disagreement between samples on a specific set

We provide three proofs of the following lemma – the constants are somewhat different for each version.

Lemma 21.1.3. *Let R_1 and R_2 be two ρ-samples from a ground set S, and consider a fixed set $T \subseteq \mathsf{S}$. We have that*

$$\mathbb{P}\Big[\big||\mathsf{R}_1 \cap T| - |\mathsf{R}_2 \cap T|\big| > \varepsilon\rho\Big] \leq 3\exp\big(-\varepsilon^2\rho/2\big).$$

Proof: (Simplest proof.) By Chernoff's inequality, for $\delta \in (0,1)$, we have

$$\mathbb{P}\Big[\big||\mathsf{R}_1| - \rho\big| \geq (\varepsilon/2)\rho\Big] \leq 2\exp\big(-(\varepsilon/2)^2\rho/4\big) = 2\exp\big(-\varepsilon^2\rho/16\big).$$

The same holds for R_2, and as such we have

$$\begin{aligned}\mathbb{P}\Big[\big||\mathsf{R}_1| - |\mathsf{R}_2|\big| \geq \varepsilon\rho\Big] &\leq \mathbb{P}\Big[\big||\mathsf{R}_1| - \rho\big| + \big|\rho - |\mathsf{R}_2|\big| \geq \varepsilon\rho\Big] \\ &\leq \mathbb{P}\Big[\big||\mathsf{R}_1| - \rho\big| \geq (\varepsilon/2)\rho\Big] + \mathbb{P}\Big[\big|\rho - |\mathsf{R}_2|\big| \geq (\varepsilon/2)\rho\Big] \leq 4\exp\big(-\varepsilon^2\rho/16\big)\end{aligned}$$ ∎

Proof: For an object $f_i \in \mathsf{S}$, let X_i be a random variable, where

$$X_i = \begin{cases} 1 & f_i \in \mathsf{R}_1 \text{ and } f_i \notin \mathsf{R}_2 \\ -1 & f_i \notin \mathsf{R}_1 \text{ and } f_i \in \mathsf{R}_2 \\ 0 & \text{otherwise.} \end{cases}$$

We have that $p_i = \mathbb{P}[X_i = 1] = \mathbb{P}[X_i = -1] = (\rho\omega_i/W)(1 - \rho\omega_i/W)$ and $\mathbb{E}[X_i] = 0$. Applying the regular concentration inequalities in this case is not immediate, since there are many X_is that are zero. To overcome this, let T be a random variable that is the number of variables in $X_1, \ldots, X_n$ that are non-zero. We have that T is a sum of n independent 0/1 random variables, where $\mathbb{E}[T] = \sum_i 2p_i = 2\rho$. In particular, by Chernoff's inequality, we have that

$$q_1 = \mathbb{P}\big[T > (1+\varepsilon)2\rho\big] \leq \exp\big(-2\rho\varepsilon^2/4\big) = \exp\big(-\rho\varepsilon^2/2\big).$$

and assume this happens. In particular, let $Z_1, \ldots, Z_T$ be the non-zero variables in $X_1, \ldots, X_n$, and observe that $\mathbb{P}[Z_i = 1] = \mathbb{P}[Z_i = -1] = 1/2$. Let $Y = \sum_i X_i = \sum_i Z_i$. Observe that $\mathbb{E}[Y] = 0$, and by Chernoff inequality, we have that

$$\begin{aligned} q_2 &= \mathbb{P}\Big[\big||\mathsf{R}_1 \cap \mathsf{S}| - |\mathsf{R}_1 \cap \mathsf{S}|\big| > \varepsilon\rho\Big] = \mathbb{P}\Big[|Y - \mathbb{E}[Y]| \geq \varepsilon\rho\Big] \leq \mathbb{P}\Big[\big|\textstyle\sum_i Z_i - 0\big| \geq \varepsilon\rho\Big] \\ &\leq 2\exp\left(-2\frac{(\varepsilon\rho)^2}{2T}\right) \leq 2\exp\left(-2\frac{(\varepsilon\rho)^2}{2(1+\varepsilon)\rho}\right) + q_1 = 2\exp\left(-\frac{\varepsilon^2\rho}{1+\varepsilon}\right) + q_1 \leq 3\exp\big(-\varepsilon^2\rho/2\big), \end{aligned}$$

using $T \leq (1+\xi)2\rho$. ∎

21.1.2. Exponential decay for a single set

Lemma 21.1.4. *Consider a set* S *of* m *objects, where every object* $f_i \in \mathsf{S}$ *has weight* $\omega_i > 0$*, and* $W = \sum_{i=1}^{m} \omega_i$*. Next, consider a set* $\mathbf{r} \subseteq \mathsf{S}$ *such that* $\omega(\mathbf{r}) \geq tW/\rho$ *(such a set is* t**-heavy***). Let* R *be a* ρ*-sample from* S*. Then, the probability that* R *misses* $\mathbf{r}$ *is at most* e^{-t}*. Formally, we have* $\mathbb{P}[\mathbf{r} \cap \mathsf{R} = \emptyset] \leq \exp(-t)$.

Proof: Let $\mathbf{r} = \{f_1, \ldots, f_k\}$. Clearly, the probability that R fails to pick one of these conflicting objects, is bounded by $\mathbb{P}[\mathbf{r} \cap \mathsf{R} = \emptyset] = \mathbb{P}\big[\forall i \in \{1, \ldots, k\} \quad f_i \notin \mathsf{R}_2\big] = \prod_{i=1}^{k}(1 - \rho\frac{\omega_i}{W}) \leq \prod_{i=1}^{k} \exp(-\rho\frac{\omega_i}{W}) = \exp(-\frac{\rho}{W}\sum_i \omega_i) \leq \exp\Big(-\frac{\rho}{W} \cdot t\frac{W}{\rho}\Big) = \exp(-t)$. ∎

21.1.3. Moments of the sample size

Lemma 21.1.5. *Let* R *an* m*-sample. And let* $f(t) \leq \alpha t^{\beta}$*, where* $\alpha \geq 1$ *and* $\beta \geq 1$ *are constants, such that* $m \geq 16\beta$*. Then* $U(m) = \mathbb{E}\big[f\big(|\mathsf{R}|\big)\big] \leq 2\alpha(2m)^{\beta}$.

Proof: The proof follows from Chernoff's inequality and some tedious but straightforward calculations. The reader is as such encouraged to skip reading it.

Let $X = |\mathsf{R}|$. This is a sum of 0/1 random variables with expectation m. As such, we have

$$\nu = \mathbb{E}\big[f\big(|\mathsf{R}|\big)\big] \leq \sum_{i=0}^{\infty} \mathbb{P}[X = i] f(i) \leq \alpha \sum_{i=0}^{\infty} \mathbb{P}[X = i] i^{\beta}.$$

Considering the last sum, we have

$$\sum_{i=0}^{\infty} \mathbb{P}\big[X = i\big] i^{\beta} \leq \sum_{j=0}^{\infty} \mathbb{P}\big[X \geq jm\big]((j+1)m)^{\beta} \leq (2m)^{\beta} + m^{\beta} \sum_{j=2}^{\infty} \mathbb{P}\big[X \geq jm\big](j+1)^{\beta}.$$

We bound the last summation using Chernoff's inequality (see Theorem 8.2.1), we have

$$\begin{aligned}
\tau &= \sum_{j=2}^{5} \mathbb{P}\big[X \geq jm\big](j+1)^{\beta} + \sum_{j=6}^{\infty} \mathbb{P}\big[X \geq jm\big](j+1)^{\beta} \\
&\leq \sum_{j=2}^{5} \exp\Big(-\frac{m(j-1)^2}{4}\Big)(j+1)^{\beta} + \sum_{j=6}^{\infty} 2^{-jm}(j+1)^{\beta} \\
&\leq \exp\Big(-\frac{m}{4}\Big)3^{\beta} + \exp(-m)4^{\beta} + \exp(-2m)5^{\beta} + \exp(-4m)6^{\beta} + \sum_{j=6}^{\infty} 2^{-jm}(j+1)^{\beta} < 1,
\end{aligned}$$

since $m \geq 16\beta$. We conclude that $\nu \leq \alpha(2m)^{\beta} + \alpha m^{\beta}\tau \leq 2\alpha(2m)^{\beta}$. ∎

Remark 21.1.6. The constant 16 in the above lemma is somewhat strange. A better constant can be derived by breaking the range of sizes into smaller intervals and using the right Chernoff inequality. Since this is somewhat tangential to the point of this write-up, we leave it as is (i.e., this constant is not critical to our discussion).

21.1.4. Growth function

The ***growth function*** $\mathcal{G}_\delta(n)$ is the maximum number of ranges in a range space with VC dimension δ, and with n elements. By Sauer's lemma, it is known that

$$\mathcal{G}_\delta(n) = \sum_{i=0}^{\delta} \binom{n}{i} \leq \sum_{i=0}^{\delta} \frac{n^i}{i!} \leq n^\delta, \tag{21.1}$$

The following is well known (the estimates are somewhat tedious to prove):

Lemma 21.1.7 ([Har11]). *For $n \geq 2\delta$ and $\delta \geq 1$, we have $\left(\frac{n}{\delta}\right)^\delta \leq \mathcal{G}_\delta(n) \leq 2\left(\frac{ne}{\delta}\right)^\delta$, where $\mathcal{G}_\delta(n) = \sum_{i=0}^{\delta} \binom{n}{i}$.*

Lemma 21.1.8. *Let R and R' be two independent m-samples from x. Assume that $m \geq \delta$. Then $\mathbb{E}\left[\mathcal{G}_\delta(|\mathsf{R}| + |\mathsf{R}'|)\right] \leq G_\delta(2m)$, where $G_\delta(2m) = 4\,(4em/\delta)^\delta$.*

Proof: We set $\alpha = 2\left(\frac{e}{\delta}\right)^\delta$, $\beta = \delta$, and $f(n) = \alpha n^\beta$. Duplicate every element in x, and let x' be the resulting set. Clearly, the size of a $2m$-sample R from x' is the same as $|\mathsf{R}| + |T|$. By Lemma 21.1.7, we have $\mathbb{E}\Big[\mathcal{G}_\delta\big(|\mathsf{R}|\big)\Big] \leq \mathbb{E}\Big[f\big(|\mathsf{R}|\big)\Big] \leq 2\alpha(4m)^\beta \leq 4\left(\frac{4em}{\delta}\right)^\delta$ The last inequality follows from Lemma 21.1.5. ∎

21.2. Proof of the ε-net theorem

Here we are working in the unweighted settings (i.e., the weight of a single element is one).

Theorem 21.2.1 (ε-net theorem, [HW87]). *Let $(X, \mathcal{R})$ be a range space of VC dimension δ, let x be a finite subset of X, and suppose that $0 < \varepsilon \leq 1$ and $\varphi < 1$. Let N be an m-sample from x (see Definition 21.1.1), where*

$$m \geq \max\left(\frac{8}{\varepsilon} \lg \frac{4}{\varphi}, \frac{16\delta}{\varepsilon} \lg \frac{16}{\varepsilon} \right). \tag{21.2}$$

Then N is an ε-net for x with probability at least $1 - \varphi$.

21.2.1. The proof

21.2.1.1. Reduction to double sampling

Let $n = |\mathsf{x}|$. Let N be the m-sample from x. Let $\mathcal{E}_1$ be the probability that N fails to be an ε-net. Namely,

$$\mathcal{E}_1 = \left\{ \exists \mathbf{r} \in \mathcal{R} \;\middle|\; |\mathbf{r} \cap \mathsf{x}| \geq \varepsilon n \quad \text{and} \quad \mathbf{r} \cap N = \emptyset \right\}.$$

(Namely, there exists a "heavy" range $\mathbf{r}$ that does not contain any point of N.) To complete the proof, we must show that $\mathbb{P}[\mathcal{E}_1] \leq \varphi$. Let T be another m-sample generated in a similar fashion to N. Let $\mathcal{E}_2$ be the event that N fails but T "works". Formally

$$\mathcal{E}_2 = \left\{ \exists \mathbf{r} \in \mathcal{R} \;\middle|\; |\mathbf{r} \cap \mathsf{x}| \geq \varepsilon n, \mathbf{r} \cap N = \emptyset, \text{ and } |\mathbf{r} \cap T| \geq \frac{\varepsilon m}{2} \right\}.$$

Intuitively, since $\mathbb{E}\big[|\mathbf{r} \cap \mathsf{x}|\big] \geq \varepsilon m$, we have that for the range $\mathbf{r}$ that N fails for, it follows with "good" probability that $|\mathbf{r} \cap T| \geq \varepsilon m/2$. Namely, $\mathcal{E}_1$ and $\mathcal{E}_2$ have more or less the same probability.

Claim 21.2.2. $\mathbb{P}[\mathcal{E}_2] \le \mathbb{P}[\mathcal{E}_1] \le 2\,\mathbb{P}[\mathcal{E}_2]$.

Proof: Clearly, $\mathcal{E}_2 \subseteq \mathcal{E}_1$, and thus $\mathbb{P}[\mathcal{E}_2] \le \mathbb{P}[\mathcal{E}_1]$. As for the other part, note that by the definition of conditional probability, we have

$$\mathbb{P}[\mathcal{E}_2 \mid \mathcal{E}_1] = \mathbb{P}[\mathcal{E}_2 \cap \mathcal{E}_1]/\mathbb{P}[\mathcal{E}_1] = \mathbb{P}[\mathcal{E}_2]/\mathbb{P}[\mathcal{E}_1].$$

It is thus enough to show that $\mathbb{P}[\mathcal{E}_2 \mid \mathcal{E}_1] \ge 1/2$.

Assume that $\mathcal{E}_1$ occurs. There is $\mathbf{r} \in \mathcal{R}$, such that $|\mathbf{r} \cap \mathsf{x}| > \varepsilon n$ and $\mathbf{r} \cap N = \emptyset$. The required probability is at least the probability that for this specific $\mathbf{r}$, we have $X = |\mathbf{r} \cap T| \ge \frac{\varepsilon n}{2}$. The variable X is a sum of $t = |\mathbf{r} \cap \mathsf{x}| \ge \varepsilon n$ random independent 0/1 variables, each one has probability m/n to be one. Setting $\mu = \mathbb{E}[X] = tm/n \ge \varepsilon m$ and $\xi = 1/2$, we have by Chernoff's inequality that

$$\mathbb{P}[|\mathbf{r} \cap T| \le \varepsilon m/2] \le \mathbb{P}[X < (1-\xi)\mu] \le \exp\bigl(-\mu\xi^2/2\bigr) = \exp(-\varepsilon m/8) < 1/2,$$

if $\varepsilon m \ge 8$. Thus, for $\mathbf{r} \in \mathcal{E}_1$, we have $\mathbb{P}[\mathcal{E}_2]/\mathbb{P}[\mathcal{E}_1] \ge \mathbb{P}\bigl[|\mathbf{r} \cap T| \ge \frac{\varepsilon m}{2}\bigr] = 1 - \mathbb{P}\bigl[|\mathbf{r} \cap T| < \frac{\varepsilon m}{2}\bigr] \ge \frac{1}{2}$. ∎

Claim 21.2.2 implies that to bound the probability of $\mathcal{E}_1$, it is enough to bound the probability of $\mathcal{E}_2$. Let

$$\mathcal{E}_2' = \left\{ \exists \mathbf{r} \in \mathcal{R} \;\middle|\; \mathbf{r} \cap N = \emptyset \text{ and } |\mathbf{r} \cap T| \ge \frac{\varepsilon m}{2} \right\}.$$

Clearly, $\mathcal{E}_2 \subseteq \mathcal{E}_2'$. Thus, bounding the probability of $\mathcal{E}_2'$ is enough to prove Theorem 21.2.1. Note, however, that a shocking thing happened! We no longer have x participating in our event. Namely, we turned bounding an event that depends on a global quantity (i.e., the ground set x) into bounding a quantity that depends only on a local quantity/experiment (involving only N and T). This is the crucial idea in this proof.

21.2.1.2. Using double sampling to finish the proof

Claim 21.2.3. $\mathbb{P}\bigl[\mathcal{E}_2\bigr] \le \mathbb{P}\bigl[\mathcal{E}_2'\bigr] \le 2^{-\varepsilon m/2} G_\delta(2m)$.

Proof: We fix the content of $\mathsf{R} = N \cup T$. The range space $(\mathsf{R}, \mathcal{R}_{|\mathsf{R}})$ has $\mathcal{G}_\delta(|\mathsf{R}|)$ ranges. Fix a range $\mathbf{r}$ in this range space. Let $T = \mathbf{r} \cap \mathsf{R}$. If $b = |T| < \varepsilon m/2$ then the $\mathcal{E}_2'$ can not happened. Otherwise, the probability that $\mathbf{r}$ is a bad range is $\mathbb{P}\Bigl[T \subseteq T \text{ and } T \cap N = \emptyset \;\Big|\; T \subseteq \mathsf{R}\Bigr] \le \frac{1}{2^b}$, by Lemma 21.1.2. In particular, by the union bound over all ranges, we have $\mathbb{P}\bigl[\mathcal{E}_2' \mid \mathsf{R}\bigr] \le 2^{-\varepsilon m/2}\mathcal{G}_\delta(|\mathsf{R}|)$. As such, we have

$$\mathbb{P}\bigl[\mathcal{E}_2'\bigr] = \sum_{\mathsf{R}} \mathbb{P}\bigl[\mathcal{E}_2' \mid \mathsf{R}\bigr]\,\mathbb{P}[\mathsf{R}] \le \sum_{\mathsf{R}} 2^{-\varepsilon m/2}\mathcal{G}_\delta\bigl(|\mathsf{R}|\bigr)\,\mathbb{P}[\mathsf{R}] \le 2^{-\varepsilon m/2}\,\mathbb{E}\Bigl[\mathcal{G}_\delta\bigl(|\mathsf{R}|\bigr)\Bigr] \le 2^{-\varepsilon m/2} G_\delta(2m).$$

by Lemma 21.1.8. ∎

Proof of Theorem 21.2.1. By Claim 21.2.2 and Claim 21.2.3, we have that $\mathbb{P}[\mathcal{E}_1] \le 2 \cdot 2^{-\varepsilon m/2} G_\delta(2m)$. It thus remains to verify that if m satisfies Eq. (21.2), then the above is smaller than φ. Which is equivalent to

$$\begin{aligned} 2 \cdot 2^{-\varepsilon m/2} G_\delta(2m) \le \varphi &\iff 16 \cdot 2^{-\varepsilon m/2}\left(\frac{4em}{\delta}\right)^{\delta} \le \varphi \iff -4 + \frac{\varepsilon m}{2} - \delta \lg\left(\frac{4em}{\delta}\right) \ge \lg\frac{1}{\varphi} \\ &\iff \left(\frac{\varepsilon m}{8} - 4 - \delta \lg\frac{4e}{\delta}\right) + \left(\frac{\varepsilon m}{8} - \lg\frac{1}{\varphi}\right) + \left(\frac{\varepsilon m}{4} - \delta \lg\left(\frac{m}{\delta}\right)\right) \ge 0 \end{aligned}$$

We remind the reader that the value of m we pick is such that $m \geq \max\left(\frac{8}{\varepsilon}\lg\frac{4}{\varphi}, \frac{16\delta}{\varepsilon}\lg\frac{16}{\varepsilon}\right)$. In particular, $m \geq 64\delta/\varepsilon$ and $-4 - \delta\lg\left(\frac{4e}{\delta}\right) \geq -4 - 4\delta \leq -8\delta \geq -\varepsilon m/8$. Similarly, by the choice of m, we have $\varepsilon m/8 \geq \lg\frac{1}{\varphi}$. As such, we need to show that $\frac{\varepsilon m}{4} \geq \delta\lg\left(\frac{m}{\delta}\right) \iff m \geq \frac{4\delta}{\varepsilon}\lg\frac{m}{\delta}$, and one can verify using some easy but tedious calculations that this holds if $m \geq \frac{16\delta}{\varepsilon}\lg\frac{16}{\varepsilon}$. ■

Chapter 22

Martingales

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
December 10, 2019

> 'After that he always chose out a "dog command" and sent them ahead. It had the task of informing the inhabitants in the village where we were going to stay overnight that no dog must be allowed to bark in the night otherwise it would be liquidated. I was also on one of those commands and when we came to a village in the region of Milevsko I got mixed up and told the mayor that every dog-owner whose dog barked in the night would be liquidated for strategic reasons. The mayor got frightened, immediately harnessed his horses and rode to headquarters to beg mercy for the whole village. They didn't let him in, the sentries nearly shot him and so he returned home, but before we got to the village everybody on his advice had tied rags round the dogs muzzles with the result that three of them went mad.'
>
> The good soldier Svejk, Jaroslav Hasek

22.1. Martingales

22.1.1. Preliminaries

Let X and Y be two random variables. Let $\rho(x,y) = \mathbb{P}[(X=x) \cap (Y=y)]$. Then,

$$\mathbb{P}[X=x \mid Y=y] = \frac{\rho(x,y)}{\mathbb{P}[Y=y]} = \frac{\rho(x,y)}{\sum_z \rho(z,y)}$$

and $\mathbb{E}\left[X \,\middle|\, Y=y\right] = \sum_x x\, \mathbb{P}\left[X=x \,\middle|\, Y=y\right] = \frac{\sum_x x\rho(x,y)}{\sum_z \rho(z,y)} = \frac{\sum_x x\rho(x,y)}{\mathbb{P}[Y=y]}$.

Definition 22.1.1. The ***conditional expectation*** of X given Y, is the random variable $\mathbb{E}\left[X \,\middle|\, Y\right]$ is the random variable $f(y) = \mathbb{E}\left[X \,\middle|\, Y=y\right]$.

Lemma 22.1.2. *For any two random variables X and Y, we have $\mathbb{E}\left[\mathbb{E}[X \mid Y]\right] = \mathbb{E}\left[X\right]$.*

Lemma 22.1.3. *For any two random variables X and Y, we have $\mathbb{E}\left[Y \cdot \mathbb{E}[X \mid Y]\right] = \mathbb{E}\left[XY\right]$.*

22.1.2. Martingales

Intuitively, martingales are a sequence of random variables describing a process, where the only thing that matters at the beginning of the ith step is where the process was in the end of the $(i-1)$th step. That is, it does not matter how the process arrived to a certain state, only that it is currently at this state.

Definition 22.1.4. A sequence of random variables $X_0, X_1, \ldots$, is said to be a ***martingale sequence*** if for all $i > 0$, we have $\mathbb{E}\left[X_i \,\middle|\, X_0, \ldots, X_{i-1}\right] = X_{i-1}$.

Lemma 22.1.5. *Let $X_0, X_1, \ldots$, be a martingale sequence. Then, for all $i \geq 0$, we have $\mathbb{E}\left[X_i\right] = \mathbb{E}\left[X_0\right]$.*

22.1.2.1. Examples of martingales

Example 22.1.6. An example of martingales is the sum of money after participating in a sequence of fair bets. That is, let X_i be the amount of money a gambler has after playing i rounds. In each round it either gains one dollar, or loses one dollar. Clearly, we have $\mathbb{E}\left[X_i \,\middle|\, X_0, \ldots, X_{i-1}\right] = \mathbb{E}\left[X_i \,\middle|\, X_{i-1}\right] = X_i$.

Example 22.1.7. Let $Y_i = X_i^2 - i$, where X_i is as defined in the above example. We claim that $Y_0, Y_1, \ldots$ is a martingale. Let us verify that this is true. Given Y_{i-1}, we have $Y_{i-1} = X_{i-1}^2 - (i-1)$. We have that

$$\begin{aligned}\mathbb{E}\left[Y_i \,\middle|\, Y_{i-1}\right] &= \mathbb{E}\left[X_i^2 - i \,\middle|\, X_{i-1}^2 - (i-1)\right] = \frac{1}{2}\Big((X_{i-1}+1)^2 - i)\Big) + \frac{1}{2}\Big((X_{i-1}-1)^2 - i\Big) \\ &= X_{i-1}^2 + 1 - i = X_{i-1}^2 - (i-1) = Y_{i-1},\end{aligned}$$

which implies that indeed it is a martingale.

Example 22.1.8. Let U be a urn with b black balls, and w white balls. We repeatedly select a ball and replace it by c balls having the same color. Let X_i be the fraction of black balls after the first i trials. We claim that the sequence $X_0, X_1, \ldots$ is a martingale.

Indeed, let $n_i = b + w + i(c-1)$ be the number of balls in the urn after the ith trial. Clearly,

$$\begin{aligned}\mathbb{E}\left[X_i \,\middle|\, X_{i-1}, \ldots, X_0\right] &= X_{i-1} \cdot \frac{(c-1) + X_{i-1} n_{i-1}}{n_i} + (1 - X_{i-1}) \cdot \frac{X_{i-1} n_{i-1}}{n_i} \\ &= \frac{X_{i-1}(c-1) + X_{i-1} n_{i-1}}{n_i} = X_{i-1} \frac{c - 1 + n_{i-1}}{n_i} = X_{i-1} \frac{n_i}{n_i} = X_{i-1}.\end{aligned}$$

Example 22.1.9. Let G be a random graph on the vertex set $V = \{1, \ldots, n\}$ obtained by independently choosing to include each possible edge with probability p. The underlying probability space is called $\mathbf{G}_{n,p}$. Arbitrarily label the $m = n(n-1)/2$ possible edges with the sequence $1, \ldots, m$. For $1 \leq j \leq m$, define the indicator random variable I_j, which takes values 1 if the edge j is present in G, and has value 0 otherwise. These indicator variables are independent and each takes value 1 with probability p.

Consider any real valued function f defined over the space of all graphs, e.g., the clique number, which is defined as being the size of the largest complete subgraph. The ***edge exposure martingale*** is defined to be the sequence of random variables $X_0, \ldots, X_m$ such that

$$X_i = \mathbb{E}\left[f(G) \,\middle|\, I_1, \ldots, I_i\right],$$

while $X_0 = \mathbb{E}[f(G)]$ and $X_m = f(G)$. This sequence of random variable begin a martingale follows immediately from a theorem that would be described in the next lecture.

One can define similarly a ***vertex exposure martingale***, where the graph G_i is the graph induced on the first i vertices of the random graph G.

Example 22.1.10 (The sheep of Mabinogion). The following is taken from medieval Welsh manuscript based on Celtic mythology:

> "And he came towards a valley, through which ran a river; and the borders of the valley were wooded, and on each side of the river were level meadows. And on one side of the river he saw a flock of white sheep, and on the other a flock of black sheep. And whenever one of the white sheep bleated, one of the black sheep would cross over and become white; and when one of the black sheep bleated, one of the white sheep would cross over and become black."
> – *Peredur the son of Evrawk*, from the *Mabinogion*.

More concretely, we start at time 0 with w_0 white sheep, and b_0 black sheep. At every iteration, a random sheep is picked, it bleats, and a sheep of the other color turns to this color. the game stops as soon as all the sheep have the same color. No sheep dies or get born during the game. Let X_i be the expected number of black sheep in the end of the game, after the ith iteration. For reasons that we would see later on, this sequence is a martingale.

The original question is somewhat more interesting – if we are allowed to take a way sheep in the end of each iteration, what is the optimal strategy to maximize X_i?

22.1.2.2. Azuma's inequality

A sequence of random variables $X_0, X_1, \ldots$ has ***bounded differences*** if $|X_i - X_{i-1}| \le \Delta$, for some Δ.

Theorem 22.1.11 (Azuma's Inequality.). *Let $X_0, \ldots, X_m$ be a martingale with $X_0 = 0$, and $|X_{i+1} - X_i| \le 1$ for all $0 \le i < m$. Let $\lambda > 0$ be arbitrary. Then $\mathbb{P}\left[X_m > \lambda\sqrt{m}\right] < \exp\left(-\lambda^2/2\right)$.*

Proof: Let $\alpha = \lambda/\sqrt{m}$. Let $Y_i = X_i - X_{i-1}$, so that $|Y_i| \le 1$ and $\mathbb{E}\left[Y_i \,\middle|\, X_0, \ldots, X_{i-1}\right] = 0$.

We are interested in bounding $\mathbb{E}\left[e^{\alpha Y_i} \,\middle|\, X_0, \ldots, X_{i-1}\right]$. Note that, for $-1 \le x \le 1$, we have

$$e^{\alpha x} \le h(x) = \frac{e^{\alpha} + e^{-\alpha}}{2} + \frac{e^{\alpha} - e^{-\alpha}}{2} x,$$

as $e^{\alpha x}$ is a convex function, $h(-1) = e^{-\alpha}$, $h(1) = e^{\alpha}$, and $h(x)$ is a linear function. Thus,

$$\begin{aligned}
\mathbb{E}\left[e^{\alpha Y_i} \,\middle|\, X_0, \ldots, X_{i-1}\right] &\le \mathbb{E}\left[h(Y_i) \,\middle|\, X_0, \ldots, X_{i-1}\right] = h\left(\mathbb{E}\left[Y_i \,\middle|\, X_0, \ldots, X_{i-1}\right]\right) \\
&= h(0) = \frac{e^{\alpha} + e^{-\alpha}}{2} \\
&= \frac{(1 + \alpha + \frac{\alpha^2}{2!} + \frac{\alpha^3}{3!} + \cdots) + (1 - \alpha + \frac{\alpha^2}{2!} - \frac{\alpha^3}{3!} + \cdots)}{2} \\
&= 1 + \frac{\alpha^2}{2} + \frac{\alpha^4}{4!} + \frac{\alpha^6}{6!} + \cdots \\
&\le 1 + \frac{1}{1!}\left(\frac{\alpha^2}{2}\right) + \frac{1}{2!}\left(\frac{\alpha^2}{2}\right)^2 + \frac{1}{3!}\left(\frac{\alpha^2}{2}\right)^3 + \cdots = \exp\left(\alpha^2/2\right),
\end{aligned}$$

as $(2i)! \ge 2^i i!$.

Hence, by Lemma 2.1.3, we have that

$$\begin{aligned}
\mathbb{E}\left[e^{\alpha X_m}\right] &= \mathbb{E}\left[\prod_{i=1}^{m} e^{\alpha Y_i}\right] = \mathbb{E}\left[\left(\prod_{i=1}^{m-1} e^{\alpha Y_i}\right) e^{\alpha Y_m}\right] \\
&= \mathbb{E}\left[\left(\prod_{i=1}^{m-1} e^{\alpha Y_i}\right) \mathbb{E}\left[e^{\alpha Y_m} \,\middle|\, X_0, \ldots, X_{m-1}\right]\right] \le e^{\alpha^2/2}\, \mathbb{E}\left[\prod_{i=1}^{m-1} e^{\alpha Y_i}\right]
\end{aligned}$$

$$\leq \exp\bigl(m\alpha^2/2\bigr).$$

Therefore, by Markov's inequality, we have

$$\begin{aligned}\mathbb{P}\bigl[X_m > \lambda\sqrt{m}\bigr] &= \mathbb{P}\Bigl[e^{\alpha X_m} > e^{\alpha\lambda\sqrt{m}}\Bigr] = \frac{\mathbb{E}\bigl[e^{\alpha X_m}\bigr]}{e^{\alpha\lambda\sqrt{m}}} = e^{m\alpha^2/2-\alpha\lambda\sqrt{m}} \\ &= \exp\bigl(m(\lambda/\sqrt{m})^2/2 - (\lambda/\sqrt{m})\lambda\sqrt{m}\bigr) = e^{-\lambda^2/2},\end{aligned}$$

implying the result. ■

Here is an alternative form.

Theorem 22.1.12 (Azuma's Inequality). *Let $X_0, \ldots, X_m$ be a martingale sequence such that and $|X_{i+1} - X_i| \leq 1$ for all $0 \leq i < m$. Let $\lambda > 0$ be arbitrary. Then $\mathbb{P}\bigl[|X_m - X_0| > \lambda\sqrt{m}\bigr] < 2\exp\bigl(-\lambda^2/2\bigr)$.*

Example 22.1.13. Let $\chi(H)$ be the chromatic number of a graph H. What is chromatic number of a random graph? How does this random variable behaves?

Consider the vertex exposure martingale, and let $X_i = \mathbb{E}\bigl[\chi(G) \bigm| G_i\bigr]$. Again, without proving it, we claim that $X_0, \ldots, X_n = X$ is a martingale, and as such, we have: $\mathbb{P}\bigl[|X_n - X_0| > \lambda\sqrt{n}\bigr] \leq e^{-\lambda^2/2}$. However, $X_0 = \mathbb{E}[\chi(G)]$, and $X_n = \mathbb{E}\bigl[\chi(G) \bigm| G_n\bigr] = \chi(G)$. Thus,

$$\mathbb{P}\bigl[|\chi(G) - \mathbb{E}[\chi(G)]| > \lambda\sqrt{n}\bigr] \leq e^{-\lambda^2/2}.$$

Namely, the chromatic number of a random graph is highly concentrated! And we do not even know what is the expectation of this variable!

22.2. Bibliographical notes

Our presentation follows [MR95].

Chapter 23

Martingales II

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
December 10, 2019

> "The Electric Monk was a labor-saving device, like a dishwasher or a video recorder. Dishwashers washed tedious dishes for you, thus saving you the bother of washing them yourself, video recorders watched tedious television for you, thus saving you the bother of looking at it yourself; Electric Monks believed things for you, thus saving you what was becoming an increasingly onerous task, that of believing all the things the world expected you to believe."
>
> Dirk Gently's Holistic Detective Agency, Douglas Adams

23.1. Filters and Martingales

Definition 23.1.1. A ***σ-field*** $(\Omega, \mathcal{F})$ consists of a sample space Ω (i.e., the atomic events) and a collection of subsets $\mathcal{F}$ satisfying the following conditions:
(A) $\emptyset \in \mathcal{F}$.
(B) $C \in \mathcal{F} \Rightarrow \overline{C} \in \mathcal{F}$.
(C) $C_1, C_2, \ldots \in \mathcal{F} \Rightarrow C_1 \cup C_2 \ldots \in \mathcal{F}$.

Definition 23.1.2. Given a σ-field $(\Omega, \mathcal{F})$, a ***probability measure*** $\mathbb{P} : \mathcal{F} \to \mathbb{R}^+$ is a function that satisfies the following conditions.
(A) $\forall A \in \mathcal{F}$, $0 \le \mathbb{P}[A] \le 1$.
(B) $\mathbb{P}\left[\Omega\right] = 1$.
(C) For mutually disjoint events $C_1, C_2, \ldots$, we have $\mathbb{P}\left[\cup_i C_i\right] = \sum_i \mathbb{P}\left[C_i\right]$.

Definition 23.1.3. A ***probability space*** $(\Omega, \mathcal{F}, \mathbb{P})$ consists of a σ-field $(\Omega, \mathcal{F})$ with a probability measure $\mathbb{P}$ defined on it.

Definition 23.1.4. Given a σ-field $(\Omega, \mathcal{F})$ with $\mathcal{F} = 2^{\Omega}$, a ***filter*** (also ***filtration***) is a nested sequence $\mathcal{F}_0 \subseteq \mathcal{F}_1 \subseteq \cdots \subseteq \mathcal{F}_n$ of subsets of 2^{Ω}, such that:
(A) $\mathcal{F}_0 = \{\emptyset, \Omega\}$.
(B) $\mathcal{F}_n = 2^{\Omega}$.
(C) For $0 \le i \le n$, $(\Omega, \mathcal{F}_i)$ is a σ-field.

Definition 23.1.5. An ***elementary event*** or ***atomic event*** is a subset of a sample space that contains only one element of Ω.

Intuitively, when we consider a probability space, we usually consider a random variable X. The value of X is a function of the elementary event that happens in the probability space. Formally, a random variable is a mapping $X : \Omega \to \mathbb{R}$. Thus, each $\mathcal{F}_i$ defines a partition of Ω into *atomic events*. This partition is getting more and more refined as we progress down the filter.

Example 23.1.6. Consider an algorithm **Alg** that uses n random bits. As such, the underlying sample space is $\Omega = \left\{ b_1 b_2 \ldots b_n \;\middle|\; b_1, \ldots, b_n \in \{0,1\} \right\}$; that is, the set of all binary strings of length n. Next, let $\mathcal{F}_i$ be the σ-field generated by the partition of Ω into the atomic events B_w, where $w \in \{0,1\}^i$; here w is the string encoding the first i random bits used by the algorithm. Specifically,

$$B_w = \left\{ wx \;\middle|\; x \in \{0,1\}^{n-i} \right\},$$

and the set of atomic events in $\mathcal{F}_i$ is $\left\{ B_w \;\middle|\; w \in \{0,1\}^i \right\}$. The set $\mathcal{F}_i$ is the closure of this set of atomic events under complement and union. In particular, we conclude that $\mathcal{F}_0, \mathcal{F}_1, \ldots, \mathcal{F}_n$ form a filter.

Definition 23.1.7. A random variable X is said to be ***$\mathcal{F}_i$-measurable*** if for each $x \in \mathbb{R}$, the event $X \le x$ is in $\mathcal{F}_i$; that is, the set $\left\{ \omega \in \Omega \;\middle|\; X(\omega) \le x \right\}$ is in $\mathcal{F}_i$.

Example 23.1.8. Let $\mathcal{F}_0, \ldots, \mathcal{F}_n$ be the filter defined in Example 23.1.6. Let X be the parity of the n bits. Clearly, $X = 1$ is a valid event only in $\mathcal{F}_n$ (why?). Namely, it is only measurable in $\mathcal{F}_n$, but not in $\mathcal{F}_i$, for $i < n$.

As such, a random variable X is $\mathcal{F}_i$-measurable, only if it is a constant on the elementary events of $\mathcal{F}_i$. This gives us a new interpretation of what a filter is – its a sequence of refinements of the underlying probability space, that is achieved by splitting the atomic events of $\mathcal{F}_i$ into smaller atomic events in $\mathcal{F}_{i+1}$. Putting it explicitly, an atomic event $\mathcal{E}$ of $\mathcal{F}_i$, is a subset of 2^Σ. As we move to $\mathcal{F}_{i+1}$ the event $\mathcal{E}$ might now be split into several atomic (and disjoint events) $\mathcal{E}_1, \ldots, \mathcal{E}_k$. Now, naturally, the atomic event that really happens is an atomic event of $\mathcal{F}_n$. As we progress down the filter, we "zoom" into this event.

Definition 23.1.9 (Conditional expectation in a filter). Let $(\Omega, \mathcal{F})$ be any σ-field, and Y any random variable that takes on distinct values on the elementary events in $\mathcal{F}$. Then $\mathbb{E}\left[X \middle| \mathcal{F} \right] = \mathbb{E}\left[X \middle| Y \right]$.

23.2. Martingales

Definition 23.2.1. A sequence of random variables $Y_1, Y_2, \ldots,$ is said to be a ***martingale difference*** sequence if for all $i \ge 0$, we have $\mathbb{E}\left[Y_i \middle| Y_1, \ldots, Y_{i-1} \right] = 0$.

Clearly, $X_1, \ldots,$ is a martingale sequence if and only if $Y_1, Y_2, \ldots,$ is a martingale difference sequence where $Y_i = X_i - X_{i-1}$.

Definition 23.2.2. A sequence of random variables $Y_1, Y_2, \ldots,$ is

$$\begin{aligned} \text{a } \textbf{\textit{super martingale}} \text{ sequence if} \quad & \forall i \quad \mathbb{E}\left[Y_i \middle| Y_1, \ldots, Y_{i-1} \right] \le Y_{i-1}, \\ \text{and a } \textbf{\textit{sub martingale}} \text{ sequence if} \quad & \forall i \quad \mathbb{E}\left[Y_i \middle| Y_1, \ldots, Y_{i-1} \right] \ge Y_{i-1}. \end{aligned}$$

23.2.1. Martingales – an alternative definition

Definition 23.2.3. Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space with a filter $\mathcal{F}_0, \mathcal{F}_1, \ldots$. Suppose that $X_0, X_1, \ldots$, are random variables such that, for all $i \geq 0$, X_i is $\mathcal{F}_i$-measurable. The sequence $X_0, \ldots, X_n$ is a ***martingale*** provided that, for all $i \geq 0$, we have $\mathbb{E}\left[X_{i+1} \mid \mathcal{F}_i\right] = X_i$.

Lemma 23.2.4. *Let $(\Omega, \mathcal{F})$ and $(\Omega, \mathcal{G})$ be two σ-fields such that $\mathcal{F} \subseteq \mathcal{G}$. Then, for any random variable X, $\mathbb{E}\Big[\mathbb{E}\big[X \,\big|\, \mathcal{G}\big] \,\Big|\, \mathcal{F}\Big] = \mathbb{E}\big[X \,\big|\, \mathcal{F}\big]$.*

Proof: $\mathbb{E}\Big[\mathbb{E}\big[X \,\big|\, \mathcal{G}\big] \,\Big|\, \mathcal{F}\Big] = \mathbb{E}\Big[\mathbb{E}\big[X \,\big|\, G = g\big] \,\Big|\, F = f\Big]$

$$
\begin{aligned}
&= \mathbb{E}\left[\frac{\sum_x x\, \mathbb{P}[X = x \cap G = g]}{\mathbb{P}[G = g]} \,\middle|\, F = f\right] = \sum_{g \in G} \frac{\frac{\sum_x x \mathbb{P}[X=x \cap G=g]}{\mathbb{P}[G=g]} \cdot \mathbb{P}[G = g \cap F = f]}{\mathbb{P}[F = f]} \\
&= \sum_{g \in \mathcal{G}, g \subseteq f} \frac{\frac{\sum_x x\, \mathbb{P}[X=x \cap G=g]}{\mathbb{P}[G=g]} \cdot \mathbb{P}[G = g \cap F = f]}{\mathbb{P}[F = f]} = \sum_{g \in \mathcal{G}, g \subseteq f} \frac{\frac{\sum_x x\, \mathbb{P}[X=x \cap G=g]}{\mathbb{P}[G=g]} \cdot \mathbb{P}[G = g]}{\mathbb{P}[F = f]} \\
&= \sum_{g \in \mathcal{G}, g \subseteq f} \frac{\sum_x x\, \mathbb{P}[X = x \cap G = g]}{\mathbb{P}[F = f]} = \frac{\sum_x x \Big(\sum_{g \in \mathcal{G}, g \subseteq f} \mathbb{P}[X = x \cap G = g]\Big)}{\mathbb{P}[F = f]} \\
&= \frac{\sum_x x\, \mathbb{P}[X = x \cap F = f]}{\mathbb{P}[F = f]} = \mathbb{E}\big[X \,\big|\, \mathcal{F}\big]. \qquad \blacksquare
\end{aligned}
$$

Theorem 23.2.5. *Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, and let $\mathcal{F}_0, \ldots, \mathcal{F}_n$ be a filter with respect to it. Let X be any random variable over this probability space and define $X_i = \mathbb{E}\big[X \,\big|\, \mathcal{F}_i\big]$ then, the sequence $X_0, \ldots, X_n$ is a martingale.*

Proof: We need to show that $\mathbb{E}\big[X_{i+1} \,\big|\, \mathcal{F}_i\big] = X_i$. Namely,

$$
\mathbb{E}[X_{i+1} \mid \mathcal{F}_i] = \mathbb{E}\Big[\mathbb{E}\big[X \,\big|\, \mathcal{F}_{i+1}\big] \,\Big|\, \mathcal{F}_i\Big] = \mathbb{E}\big[X \,\big|\, \mathcal{F}_i\big] = X_i,
$$

by Lemma 23.2.4 and by definition of X_i. $\blacksquare$

Definition 23.2.6. Let $f : \mathcal{D}_1 \times \cdots \times \mathcal{D}_n \to \mathbb{R}$ be a real-valued function with a arguments from possibly distinct domains. The function f is said to satisfy the ***Lipschitz condition*** if for any $x_1 \in \mathcal{D}_1, \ldots, x_n \in \mathcal{D}_n$, and $i \in \{1, \ldots, n\}$ and any $y_i \in \mathcal{D}_i$,

$$
\Big| f(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n) - f(x_1, \ldots, x_{i-1}, y_i, x_{i+1}, \ldots, x_n) \Big| \leq 1.
$$

Specifically, a function is ***c-Lipschitz***, if the inequality holds with a constant c (instead of 1).

Definition 23.2.7. Let $X_1, \ldots, X_n$ be a sequence of *independent* random variables, and a function $f(X_1, \ldots, X_n)$ defined over them that such that f satisfies the Lipschitz condition. The ***Doob martingale*** sequence $Y_0, \ldots, Y_m$ is defined by $Y_0 = \mathbb{E}\big[f(X_1, \ldots, X_n)\big]$ and $Y_i = \mathbb{E}\big[f(X_1, \ldots, X_n) \,\big|\, X_1, \ldots, X_i\big]$, for $i = 1, \ldots, n$.

Clearly, a Doob martingale $Y_0, \ldots, Y_n$ is a martingale, by Theorem 23.2.5. Furthermore, if $|X_i - X_{i-1}| \leq 1$, for $i = 1, \ldots, n$, then $|Y_i - Y_{i-1}| \leq 1$. and we can use Azuma's inequality on such a sequence.

23.3. Occupancy Revisited

We have m balls thrown independently and uniformly into n bins. Let Z denote the number of bins that remains empty in the end of the process. Let X_i be the bin chosen in the ith trial, and let $Z = F(X_1, \ldots, X_m)$, where F returns the number of empty bins given that m balls had thrown into bins $X_1, \ldots, X_m$. Clearly, we have by Azuma's inequality that $\mathbb{P}\left[\left|Z - \mathbb{E}[Z]\right| > \lambda\sqrt{m}\right] \leq 2e^{-\lambda^2/2}$.

The following is an extension of Azuma's inequality shown in class. We do not provide a proof but it is similar to what we saw.

Theorem 23.3.1 (Azuma's Inequality - Stronger Form). *Let $X_0, X_1, \ldots,$ be a martingale sequence such that for each k, $|X_k - X_{k-1}| \leq c_k$, where c_k may depend on k. Then, for all $t \geq 0$, and any $\lambda > 0$, we have*

$$\mathbb{P}\Big[|X_t - X_0| \geq \lambda\Big] \leq 2\exp\left(-\frac{\lambda^2}{2\sum_{k=1}^{t} c_k^2}\right).$$

Theorem 23.3.2. *Let $r = m/n$, and Z_{end} be the number of empty bins when m balls are thrown randomly into n bins. Then $\mu = \mathbb{E}\Big[Z_{end}\Big] = n\big(1 - \frac{1}{n}\big)^m \approx ne^{-r}$, and for any $\lambda > 0$, we have*

$$\mathbb{P}\left[\left|Z_{end} - \mu\right| \geq \lambda\right] \leq 2\exp\left(-\frac{\lambda^2(n - 1/2)}{n^2 - \mu^2}\right).$$

Proof: Let $z(Y, t)$ be the expected number of empty bins, if there are Y empty bins in time t. Clearly,

$$z(Y, t) = Y\left(1 - \frac{1}{n}\right)^{m-t}.$$

In particular, $\mu = z(n, 0) = n\big(1 - \frac{1}{n}\big)^m$.

Let $\mathcal{F}_t$ be the σ-field generated by the bins chosen in the first t steps. Let Z_{end} be the number of empty bins at time m, and let $Z_t = \mathbb{E}\big[Z_{\text{end}} \,\big|\, \mathcal{F}_t\big]$. Namely, Z_t is the expected number of empty bins after we know where the first t balls had been placed. The random variables $Z_0, Z_1, \ldots, Z_m$ form a martingale. Let Y_t be the number of empty bins after t balls where thrown. We have $Z_{t-1} = z(Y_{t-1}, t-1)$. Consider the ball thrown in the t-step. Clearly:

(A) With probability $1 - Y_{t-1}/n$ the ball falls into a non-empty bin. Then $Y_t = Y_{t-1}$, and $Z_t = z(Y_{t-1}, t)$. Thus,

$$\Delta_t = Z_t - Z_{t-1} = z(Y_{t-1}, t) - z(Y_{t-1}, t-1) = Y_{t-1}\left(\left(1 - \frac{1}{n}\right)^{m-t} - \left(1 - \frac{1}{n}\right)^{m-t+1}\right) = \frac{Y_{t-1}}{n}\left(1 - \frac{1}{n}\right)^{m-t} \leq \left(1 - \frac{1}{n}\right)^{m-t}.$$

(B) Otherwise, with probability Y_{t-1}/n the ball falls into an empty bin, and $Y_t = Y_{t-1} - 1$. Namely, $Z_t = z(Y_t - 1, t)$. And we have that

$$\begin{aligned}
\Delta_t = Z_t - Z_{t-1} &= z(Y_{t-1} - 1, t) - z(Y_{t-1}, t-1) = (Y_{t-1} - 1)\left(1 - \frac{1}{n}\right)^{m-t} - Y_{t-1}\left(1 - \frac{1}{n}\right)^{m-t+1} \\
&= \left(1 - \frac{1}{n}\right)^{m-t}\left(Y_{t-1} - 1 - Y_{t-1}\left(1 - \frac{1}{n}\right)\right) = \left(1 - \frac{1}{n}\right)^{m-t}\left(-1 + \frac{Y_{t-1}}{n}\right) = -\left(1 - \frac{1}{n}\right)^{m-t}\left(1 - \frac{Y_{t-1}}{n}\right) \\
&\geq -\left(1 - \frac{1}{n}\right)^{m-t}.
\end{aligned}$$

Thus, $Z_0, \ldots, Z_m$ is a martingale sequence, where $|Z_t - Z_{t-1}| \le |\Delta_t| \le c_t$, where $c_t = \left(1 - \frac{1}{n}\right)^{m-t}$. We have

$$\sum_{t=1}^{n} c_t^2 = \frac{1 - (1-1/n)^{2m}}{1 - (1-1/n)^2} = \frac{n^2\left(1 - (1-1/n)^{2m}\right)}{2n-1} = \frac{n^2 - \mu^2}{2n-1}.$$

Now, deploying Azuma's inequality, yield the result. ∎

23.3.1. Lets verify this is indeed an improvement

Consider the case where $m = n \ln n$. Then, $\mu = n\left(1 - \frac{1}{n}\right)^m \le 1$. And using the "weak" Azuma's inequality implies that

$$\mathbb{P}\left[\left|Z_{\text{end}} - \mu\right| \ge \lambda\sqrt{n}\right] = \mathbb{P}\left[\left|Z_{\text{end}} - \mu\right| \ge \lambda\sqrt{\frac{n}{m}}\sqrt{m}\right] \le 2\exp\left(-\frac{\lambda^2 n}{2m}\right) = 2\exp\left(-\frac{\lambda^2}{2\ln n}\right),$$

which is interesting only if $\lambda > \sqrt{2\ln n}$. On the other hand, Theorem 23.3.2 implies that

$$\mathbb{P}\left[\left|Z_{\text{end}} - \mu\right| \ge \lambda\sqrt{n}\right] \le 2\exp\left(-\frac{\lambda^2 n(n-1/2)}{n^2 - \mu^2}\right) \le 2\exp\left(-\lambda^2\right),$$

which is interesting for any $\lambda \ge 1$ (say).

23.4. Some useful estimates

Lemma 23.4.1. *For any $n \ge 2$, and $m \ge 1$, we have that $(1 - 1/n)^m \ge 1 - m/n$.*

Proof: Follows by induction. Indeed, for $m = 1$ the claim is immediate. For $m \ge 2$, we have

$$\left(1 - \frac{1}{n}\right)^m = \left(1 - \frac{1}{n}\right)\left(1 - \frac{1}{n}\right)^{m-1} \ge \left(1 - \frac{1}{n}\right)\left(1 - \frac{m-1}{n}\right) \ge 1 - \frac{m}{n}.$$ ∎

This implies the following.

Lemma 23.4.2. *For any $m \le n$, we have that $1 - m/n \le (1 - 1/n)^m \le \exp(-m/n)$.*

Chapter 24

The power of two choices

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
December 10, 2019

> The Peace of Olivia. How sweat and peaceful it sounds! There the great powers noticed for the first time that the land of the Poles lends itself admirably to partition.
>
> The tin drum, Gunter Grass

Consider the problem of throwing n balls into n bins. It is well known that the maximum load is $\Theta(\log n/\log\log n)$ with high probability. Here we show that if one is allowed to pick d bins for each ball, and throw it into the bin that contains less balls, then the maximum load of a bin decreases to $\Theta(\log\log n/\log d)$. A variant of this approach leads to maximum load $\Theta((\log\log n)/d)$.

As a concrete example, for $n = 10^9$, this leads to maximum load 13 in the regular case, compared to maximum load of 4, with only two-choices – see Figure 24.1.

24.1. Balls and bins with many rows

Consider throwing n balls into n bins. Every bin can contain a single ball. As such, as we throw the balls, some balls would be rejected because their assigned bin already contains a ball. We collect all the rejected balls, and throw them again into a second row of n bins. We repeat this process till all the balls had found a good home (i.e., empty bin). How many rows one needs before this process is completed?

Lemma 24.1.1. *Let $m = \alpha n$ balls be thrown into n bins. Let Y_{end} the number of bins that are not empty in the end of the process (here, we allow more than one ball into a bin).*

(A) For $\alpha \in (0,1]$, we have $\mu = \mathbb{E}[Y_{\mathrm{end}}] \geq m\exp(-\alpha)$.

(B) If $\alpha \geq 1$, then $\mu = \mathbb{E}[Y_{\mathrm{end}}] \geq n(1-\exp(-\alpha))$.

(C) We have $\mathbb{P}\left[|Y_{\mathrm{end}} - \mu| > \sqrt{3cm\log n}\right] \leq 1/n^c$.

Proof: (A) The probability of the ith ball to be the first ball in its bin, is $\left(1-\frac{1}{n}\right)^{i-1}$. To see this we use backward analysis – throw in the ith ball, and now throw in the earlier $i-1$ balls. The probability that none of the earlier balls hit the same bin as the ith ball is as stated. Now, the expected number of non-empty bins is the number of balls that are first in their bins, which in turn is

$$\mu = \sum_{i=0}^{m-1}\left(1-\frac{1}{n}\right)^i \geq m(1-1/n)^m \geq m(1-1/n)^{(n-1)m/(n-1)} \geq m\exp\left(-\frac{m}{n-1}\right) = m\exp\left(-\frac{\alpha(n-1)+\alpha}{n-1}\right)$$

$$= m \exp\left(-\alpha + \frac{\alpha}{n-1}\right) \geq m \exp(-\alpha) \geq \frac{m}{e}.$$

using $m = \alpha n \leq n$, and $(1 - 1/n)^{n-1} \geq 1/e$, see Lemma 6.1.8.

(B) We repeat the above analysis from the point of view of the bin. The probability of a bin to be empty is $(1 - 1/n)^{\alpha n}$. As such, we have that

$$\mu = \mathbb{E}[Y_{\text{end}}] = n(1 - (1 - 1/n)^{\alpha n}) \geq n(1 - \exp(-\alpha)),$$

using $1 - 1/n \leq \exp(-1/n)$.

(C) Let X_i be the index of the bin the ith ball picked. Let $Y_i = \mathbb{E}[Y_{\text{end}} \mid X_1, \ldots, X_i]$. This is a Doob martingale, with $|Y_i - Y_{i-1}| \leq 1$. As such, Azuma's inequality implies, for $\lambda = \sqrt{3cm \ln n}$, that

$$\mathbb{P}\left[|Y_{\text{end}} - \mathbb{E}[Y_{\text{end}}]| \geq \lambda\right] \leq 2 \exp\left(-\lambda^2/2m\right) \leq 1/n^c. \qquad \blacksquare$$

Remark. The reader might be confused by cases (A) and (B) of Lemma 24.1.1 for $\alpha = 1$, as the two lower bounds are different. Observe that (A) is loose if α is relatively large and close to 1.

Back to the problem. Let $\alpha_1 = 1$ and $n_1 = \alpha_1 n$. For $i > 1$, inductively, assume that numbers of balls being thrown in the ith round is

$$n_i = \alpha_i n + O(\sqrt{\alpha_{i-1} n \log n}).$$

By Lemma 24.1.1, with high probability, the number of balls stored in the ith row is

$$s_i = n_i \exp(-\alpha_i) \pm O(\sqrt{n_i \log n}).$$

As such, as long as the first term is significantly large than the second therm, we have that $s_i = n\alpha_i \exp(-\alpha_i)(1 \pm o(1))$. For the time being, let us ignore the $o(1)$ term. We have that

$$n_{i+1} = n_i - s_i = n(\alpha_i - \alpha_i \exp(-\alpha_i)) \leq n(\alpha_i - \alpha_i(1 - \alpha_i)) = n\alpha_i^2,$$

since $\exp(-\alpha_i) \geq 1 - \alpha_i$.

Observation 24.1.3. *Consider the sequence $\alpha_1 = 1$, $c = \alpha_2 = 1 - 1/e$, and $\alpha_{i+1} = \alpha_i^2$, for $i > 2$. We have that $\alpha_{i+1} = c^{2^{i-2}}$. In particular, for $\Delta = 3 + \lg \log_{1/c} n$, we have that $\alpha_\Delta < 1/n$.*

The above observation almost implies that we need Δ rows. The problem is that the above calculations (i.e., the high probability guarantee in Lemma 24.1.1) breaks down when $n_i = O(\log n)$ – that is, when $\alpha_i = O((\log n)/n)$. However, if one throws in $O(\log n)$ balls into n bins, the probability of a single collision is at most $O((\log n)^2/n)$. In particular, this implies that after roughly additional c rows, the probability of any ball left is $\leq 1/n^c$.

The above argumentation, done more carefully, implies the following – we omit the details because (essentially) the same analysis for a more involved case is done next (the lower bound stated follows also from the same argumentation).

Theorem 24.1.4. *Consider the process of throwing n balls into n bins in several rounds. Here, a ball that can not be placed in a round, because their chosen bin is already occupied, are promoted to the next round. The next round throws all the rejected balls from the previous round into a new row of n empty bins. This process, with high probability, ends after $M = \lg \lg n + \Theta(1)$ rounds (i.e., after M rounds, all balls are placed in bins).*

24.1.1. With only d rows

Lemma 24.1.5. *For $\alpha \in (0, 1/4]$, let $\gamma_1 = \alpha$, and $\gamma_i = 2\gamma_{i-1}^2$. We have that $\gamma_{d+1} \leq \alpha^{(2^d+1)/2}$.*

Proof: The proof, minimal as it may be, is by induction:

$$\gamma_{i+1} = 2\gamma_i^2 \leq 2\left(\alpha^{(2^{i-1}+1)/2}\right)^2 = 2\alpha^{(2^i+2)/2} \leq \alpha^{(2^i+1)/2},$$

since $2\sqrt{\alpha} \leq 1$. ∎

Lemma 24.1.6. *Let $m = \alpha n$ balls be thrown into n bins, with d rows, where $\alpha > 0$. Here every bin can contain only a single ball, and if inserting the ball into ith row failed, then we throw it in the next row, and so on, till it finds an empty bin, or it is rejected because it failed on the dth row. Let $Y(d, n, m)$ be the number of balls that did not get stored in this matrix of bins. We have*

(A) For a constant $\alpha < 1/4$, we have $Y(d, n, \alpha n) \leq n\alpha^{(2^d+1)/2}$, with high probability.

(B) We $\mathbb{E}\big[Y(d, n, dn)\big] = O(n \log d)$.

(C) For a constant $c > 1$, we have $\mathbb{E}\big[Y(d, n, cn \log d)\big] = n/e^{-d/2}$, assuming d is sufficiently large.

Proof: (A) By Lemma 24.1.1, in expectation, at least $s_1 = n\alpha \exp(-\alpha)$ balls are placed in the first row. As such, in expectation $n_2 = n\alpha(1 - \exp(-\alpha)) \leq n\alpha^2$ balls get thrown into the second row. Using Chenroff inequality, we get that $n_2 \leq 2\alpha^2 n$, with high probability. Setting $\gamma_1 = \alpha$, and $\gamma_i = 2\gamma_{i-1}^2$, we get the claim via Lemma 24.1.5.

(B) As long as we throw $\Omega(n \log d)$ balls into a row, we expect by Lemma 24.1.1 that at least $n(1 - 1/d^{O(1)})$ balls to get stored in this row. As such, let $D = O(\log d)$, and observe that the first $d - D$ rows in expectation contains $n(d - D)(1 - 1/d^{O(1)})$ balls. This implies that only $O(Dn)$ are not stored in these first $d - D$ rows, which implies the claim.

(C) Break the d rows into two groups. The first group of size $D = O(\log d)$, and the second group is the remaining group. As long as the number of balls arriving to a row is larger than n, we expected at least $n(1 - 1/e)$ of them to be stored in this row. As such, after the first D rows, we expect the number of remaining balls to be $\leq n$. But them. the same argumentation implies that the number of balls arriving to the $D + i$ row, in expectation, is at most n/e^i. In particular, we get that the number of balls failed to be placed is at most $n/e^{D-d} \leq n/e^{-d/2}$. ∎

24.2. The power of two choices

Making d choices. Let us throw n balls into n bins. For each ball, we first pick randomly $d \geq 2$ bins, and place the ball in the bin that currently contains the smallest number of balls (here, a bin might contain an arbitrary number of balls). If there are several bins with the minimum number of bins, we resolve it arbitrarily.

Here, we will show the surprising result that the maximum number of balls in any bin is bounded by $O(\log \log n / \log d)$ with high probability in the end of this process. For $d = 1$, which is the regular balls into bins setting, we already seen that this quantity is $\Theta(\log n / \log \log n)$, so this result is quite surprising.

24.2.1. Upper bound

Definition 24.2.1. The ***load*** of a bin is the number of balls in it. The ***height*** of a ball, is the load of the bin it was inserted into, just after it was inserted.

Some notations:

(A) β_i: An upper bound on the number of bins that have load at least i by the end of the process.
(B) $h(i)$: The height of the ith ball.
(C) $\sqcup_{\geq i}(t)$: Number of bins with load at least i at time t.
(D) $\text{⚽}_{\geq i}(t)$: Number of balls with height at least i at time t.

Observation 24.2.2. $\sqcup_{\geq i}(t) \leq \text{⚽}_{\geq i}(t)$.

Let $⊎_{\geq i} = \sqcup_{\geq i}(n)$ be the number of bins, in the end of the process, that have load $\geq i$.

Observation 24.2.3. *Since every bin counted in $⊎_{\geq i}$ contains at least i balls, and there are n balls, it follows that $⊎_{\geq i} \leq n/i$.*

Lemma 24.2.4. *Let $\beta_4 = n/4$, and let $\beta_{i+1} = 2n(\beta_i/n)^d$, for $i \geq 4$. Let I be the last iteration, such that $\beta_I \geq 16c \ln n$, where $c > 1$ is an arbitrary constant. Then, with probability $\geq 1 - 1/n^c$, we have that*

(A) $⊎_{\geq i} \leq \beta_i$, for $i = 4, \ldots, I$.
(B) $⊎_{\geq I+1} \leq c' \log n$, for some constant c'.
(C) For $j > 0$, and any constant $\varepsilon > 0$, we have $\mathbb{P}\left[⊎_{\geq I+1+j} > 0\right] \leq O(1/n^{(d-1-\varepsilon)j})$.
(D) With probability $\geq 1 - 1/n^c$, the maximum load of a bin is $I + O(c)$.

Proof: (A) Let $\mathcal{B}_i$ be the bad event that $⊎_{\geq i} > \beta_i$, for $i = 1, \ldots, n$. The following analysis is conditioned on none of these bad events happening. Let $\mathcal{G}$ be the good event that is the complement of $\cup_i \mathcal{B}_i$. Let Y_t be an indicator variable that is one $\iff$ $h(t) \geq i+1$ conditioned on $\mathcal{G}$ (for clarity, we omit mentioning this conditioning explicitly). We have that

$$\tau_j = \mathbb{P}\left[Y_j = 1\right] \leq p_i \qquad \text{for} \qquad p_i = (\beta_i/n)^d,$$

as all d probes must hit bins of height at least i, and there are at most β_i such bins. This readily implies that $\mathbb{E}[\text{⚽}_{\geq i+1}(n)] \leq p_i n$. The variables $Y_1, \ldots, Y_n$ are not independent, but consider a variable Y'_j that is 1 if $Y_j = 1$, or if $Y_j = 0$, then Y'_j is 1 with probability $p - \tau_j$. Clearly, the variables $Y'_1, \ldots, Y'_n$ are independent, and $\sum_i Y'_j \geq \sum_i Y_i$. For $i < I$, setting

$$\beta_{i+1} = 2np_i = 2n(\beta_i/n)^d,$$

we have, by Chernoff's inequality, that

$$\alpha_{i+1} = \mathbb{P}\left[\mathcal{B}_{i+1} \;\middle|\; \cap_{k=1}^{i} \overline{\mathcal{B}_1}\right] = \mathbb{P}[\text{⚽}_{\geq i+1}(n) > \beta_{i+1}] = \mathbb{P}[\text{⚽}_{\geq i+1}(n) > 2np_i] \leq \mathbb{P}\left[\sum_i Y'_t > (1+1)np_i\right]$$
$$\leq \exp(-np_i/4) = \exp(-\beta_{i+1}/8) < 1/n^{2c}.$$

(B) For β_{I+1}, we have $\beta_{I+1} \leq 16c \log n$. Setting $\Delta = 2e \cdot 16c \log n$, we have

$$\alpha_{I+1} = \mathbb{P}[\text{⚽}_{\geq I+1}(n) > \beta_{I+1}] \leq \mathbb{P}\left[\sum_i Y'_t > \frac{\Delta}{\beta_{I+1}}\beta_{I+1}\right] \leq 2^{-32ec \log n} \leq \frac{1}{n^c},$$

by Lemma 8.2.7.

As for the conditioning used in the above, we have that $\mathbb{P}[\mathcal{G}] = \prod_{\ell=4}^{I+1} \mathbb{P}\left[\mathcal{B}_{\ell+1} \;\middle|\; \cap_{k=1}^{\ell} \overline{\mathcal{B}_1}\right] = \prod_i (1-\alpha_i) \geq 1 - 1/n^{c-1}$, since $I \leq n$.

(C) Observe that $\sqcup_{\geq i+1}(n) \leq \sqcup_{\geq i}(n)$. As such, for all $j > 0$, we have that $\sqcup_{\geq I+1+j}(n) \leq \text{⚽}_{\geq I+1}(n) \leq \Delta = 2e \cdot 16c \log n$, by (B). As such, we have

$$\mathbb{E}\left[\text{⚽}_{\geq I+1+j}(n)\right] \leq n(\Delta/n)^d = O(\log^d n / n^{d-1}) = O(1/n^{d-1-\varepsilon}) \ll 1,$$

for $\varepsilon > 0$ an arbitrary constant, and n sufficient large. Using Markov's inequality, we get that $q = \mathbb{P}\left[\text{⚽}_{\geq I+1+j}(n) \geq 1\right] = O(1/n^{d-1-\varepsilon})$. The probability that the first j such rounds fail (i.e., that $\text{⚽}_{\geq I+1+j}(n) > 0$) is at most q^j, as claimed.

(D) This follows immediately by picking $\varepsilon = 1/2$, and then using (C) with $j = O(c)$. ∎

Lemma 24.2.5. *For $i = 4, \ldots, I$, we have that $\beta_i \leq n/2^{d^{i-4}+1}$.*

Proof: The proof is by induction. For $i = 4$, we have $\beta_4 \leq n/4$, as claimed. Otherwise, we have

$$\beta_{i+1} = 2n(\beta_i/n)^d \leq 2n\left(1/2^{d^{i-4}+1}\right)^d = n/2^{d^{i+1-4}+d-1} \leq n/2^{d^{i+1-4}+1}.$$ ∎

Theorem 24.2.6. *When throwing n balls into n bins, with d choices, with probability $\geq 1 - 1/n^{O(1)}$, we have that the maximum load of a bin is $O(1) + \lg\lg n / \lg d$*

Proof: By Lemma 24.2.4, with the desired probability the β_is bound the load in the bins for $i \leq I$. By Lemma 24.2.5, it follows that for $I = O(1) + (\lg\lg n)/\lg d$, we have that $\beta_I \leq o(\log n)$. Thus giving us the desired bound. ∎

It is not hard to verify that our upper bounds (i.e., β_i) are not too badly off, and as such the maximum load in the worst case is (up to additive constant) the same. We state the result without proof.

Theorem 24.2.7. *When throwing n balls into n bins, with d choices (where the ball is placed with the bin with the least load), with probability $\geq 1 - o(1/n)$, we have that the maximum load of a bin is at least $\lg\lg n / \lg d - O(1)$.*

24.2.2. Applications

As a direct application, we can use this approach for open hashing, where we use two hash functions, and place an element in the bucket of the hash table with fewer elements. By the above, this improves the worst case search time from $O(\log n / \log\log n)$ to $O(\log\log n)$. This comes at the cost of doubling the time it takes to do lookup on average.

# balls in bin	Regular	2-choices	2-choices+go left
0	369,899,815	240,525,897	228,976,604
1	365,902,266	528,332,061	546,613,797
2	182,901,437	221,765,420	219,842,639
3	61,604,865	9,369,389	4,566,915
4	15,760,559	7,233	45
5	3,262,678		
6	568,919		
7	86,265		
8	11,685		
9	1,347		
10	143		
11	17		
12	2		
13	2		

Figure 24.1: Simulation of the three schemes described here. This was done with $n = 1,000,000,000$ balls thrown into n bins. Since $\log\log n$ is so small (i.e., ≈ 3 in this case, there does not seem to be any reasonable cases where the is a significant differences between d-choices and the go-left variant. In the simulations, the *go-left* variant always has a somewhat better distribution, as shown above.

24.2.3. The power of restricted d choices: Always go left

The always go left rule. Consider throwing a ball into n bins (which might already have some balls in them) as follows – you pick uniformly a number $X_i \in [\![n/d]\!]$, and you try locations $Y_1, \ldots, Y_d$, where $Y_j = X_j + j(n/d)$, for $j = 1, \ldots, d$. Let L_j be the load of bin Y_j, for $j = 1, \ldots, d$, and let $L = \min_j L_j$ be the minimum load of any bin. Let τ be the minimum index such that $L_j = L$. We throw the ball into Y_τ.

What the above scheme does, is to partition the n bins into d groups, placed from left to right. We pick a bin uniformly from each group, and always throw the ball in the leftmost location that realizes the minimum load.

The following proof is informal for the sake of simplicity.

Theorem 24.2.8. *When throwing n balls into n bins, using the always-go-left rule, with d groups of size n/d, the maximum load of a bin is $O(1) + (\log\log n)/d$, with high probability.*

Proof: Lemma 24.1.6 (B) tells us that $n(1 - O(\log d/d))$ balls get placed as the first ball in their bin, and their height is one.

Lemma 24.1.6 (C) implies that at most $dn/e^{-d/2}$ balls have height larger than 2.

Lemma 24.1.6 (A) implies that now we can repeat the same analysis as the power of two choices, the critical difference is that every one of the d groups, behaves like a separate height. Since there are $O(\log\log n)$ maximum height in the regular analysis, this implies that we get $O((\log\log n)/d)$ maximum load, with high probability. ∎

24.3. Avoiding terrible choices

Interestingly, one can prove that two choices are not really necessary. Indeed, consider the variant where the ith ball randomly chooses a random location r_i. The ball then is placed in the bin with least load among the bins r_i and r_{i-1} (the first ball inspects only a single bin – r_1). It is not difficult to show that the above analysis applies in this settings, and the maximum load is $O(\log \log n)$ – despite making only n choices for n balls. Intuitively, what is going on is that the power of two choices lies in the ability to avoid following a horrible, no good, terrible choice, by having an alternative. This alternative choice does not have to be quite of the same quality as the original choice - it can be stolen from the previous ball, etc.

24.4. Bibliographical notes

The multi-row balls into bins (Section 24.1) is from the work by Broder and Karlin [BK90]. The power of two choices (Section 24.2) is from Azar *et al.* [ABKU99].

The restricted d choices structure, the always go-left rule, described in Section 24.2.3, is from [V03].

Chapter 25

Finite Metric Spaces and Partitions

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
December 10, 2019

25.1. Finite Metric Spaces

Definition 25.1.1. A ***metric space*** is a pair $(\mathcal{X}, \mathsf{d})$ where $\mathcal{X}$ is a set and $\mathsf{d} : \mathcal{X} \times \mathcal{X} \to [0, \infty)$ is a ***metric***, satisfying the following axioms:

(i) $\mathsf{d}(x, y) = 0 \iff x = y$,
(ii) $\mathsf{d}(x, y) = \mathsf{d}(y, x)$, and
(iii) $\mathsf{d}(x, y) + \mathsf{d}(y, z) \geq \mathsf{d}(x, z)$ (triangle inequality).

The plane, $\mathbb{R}^2$, with the regular Euclidean distance is a metric space.

Of special interest is the finite case, where $\mathcal{X}$ is an n-point set. Then, the function d can be specified by $\binom{n}{2}$ real numbers. Alternatively, one can think about $(\mathcal{X}, \mathsf{d})$ as a weighted complete graph, where positive weights are specified on the edges, and these weights comply with the triangle inequality.

Finite metric spaces rise naturally from (sparse) graphs. Indeed, let $\mathsf{G} = (\mathcal{X}, \mathsf{E})$ be an undirected weighted graph defined over $\mathcal{X}$, and let $\mathsf{d}_{\mathsf{G}}(x, y)$ be the length of the shortest path between x and y in G. It is easy to verify that $(\mathcal{X}, \mathsf{d}_G)$ is a finite metric space. As such if the graph G is sparse, it provides a compact representation to the finite space $(\mathcal{X}, \mathsf{d}_G)$.

Definition 25.1.2. Let $(\mathcal{X}, d)$ be an n-point metric space. We denote the ***open ball*** of radius r about $x \in \mathcal{X}$, by $\mathbf{b}(x, r) = \{y \in \mathcal{X} \mid \mathsf{d}(x, y) < r\}$.

Underling our discussion of metric spaces are algorithmic applications. The hardness of various computational problems depends heavily on the structure of the finite metric space. Thus, given a finite metric space, and a computational task, it is natural to try to map the given metric space into a new metric where the task at hand becomes easy.

Example 25.1.3. Computing the diameter of a point set is not trivial in two dimensions (if one wants near linear running time), but is easy in one dimension. Thus, if we could map points in two dimensions into points in one dimension, such that the diameter is preserved, then computing the diameter becomes easy. This approach yields an efficient approximation algorithm, see Exercise 25.7.3 below.

Of course, this mapping from one metric space to another, is going to introduce error. Naturally, one would like to minimize the error introduced by such a mapping.

Definition 25.1.4. Let $(\mathcal{X}, \mathsf{d}_{\mathcal{X}})$ and $(\mathcal{Y}, \mathsf{d}_{\mathcal{Y}})$ be two metric spaces. A mapping $f : \mathcal{X} \to \mathcal{Y}$ is an ***embedding***, and it is ***C-Lipschitz*** if $\mathsf{d}_Y\big(f(x), f(y)\big) \leq C \cdot \mathsf{d}_{\mathcal{X}}(x, y)$ for all $x, y \in \mathcal{X}$. The mapping f is ***K-bi-Lipschitz*** if there exists a $C > 0$ such that

$$CK^{-1} \cdot \mathsf{d}_{\mathcal{X}}(x, y) \leq \mathsf{d}_Y\big(f(x), f(y)\big) \leq C \cdot \mathsf{d}_{\mathcal{X}}(x, y),$$

for all $x, y \in \mathcal{X}$.

The least K for which f is K-bi-Lipschitz is the ***distortion*** of f, and is denoted $\operatorname{dist}(f)$. The least distortion with which $\mathcal{X}$ may be embedded in $\mathcal{Y}$ is denoted $c_{\mathcal{Y}}(\mathcal{X})$.

Informally, if $f : \mathcal{X} \to \mathcal{Y}$ has distortion K, then the distances in $\mathcal{X}$ and $f(\mathcal{X}) \subseteq \mathcal{Y}$ are the same up to a factor of K (one might need to scale up the distances by some constant C).

There are several powerful results about low distortion embeddings that would be presented:

(I) **Probabilistic trees**. Every finite metric can be randomly embedded into a tree such that the "expected" distortion for a specific pair of points is $O(\log n)$.

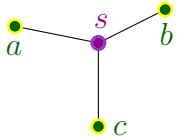
(II) **Bourgain embedding**. Any n-point metric space can be embedded into (finite dimensional) euclidean metric space with $O(\log n)$ distortion.

(III) **Johnson-Lindenstrauss lemma**. Any n-point set in Euclidean space with the regular Euclidean distance can be embedded into $\mathbb{R}^k$ with distortion $(1 + \varepsilon)$, where $k = O(\varepsilon^{-2} \log n)$.

25.2. Examples

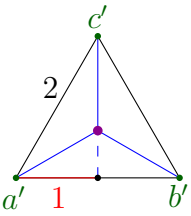
What is distortion? When considering a mapping $f : \mathcal{X} \to \mathbb{R}^d$ of a metric space $(\mathcal{X}, \mathsf{d})$ to $\mathbb{R}^d$, it would useful to observe that since $\mathbb{R}^d$ can be scaled, we can consider f to be an expansion (i.e., no distances shrink). Furthermore, we can assume that there is at least one pair of points $x, y \in \mathcal{X}$, such that $\mathsf{d}(x, y) = \|x - y\|$. As such, we have $\operatorname{dist}(f) = \max_{x,y} \frac{\|x-y\|}{\mathsf{d}(x,y)}$.

Why is distortion necessary? Consider the a graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ with one vertex s connected to three other vertices a, b, c, where the weights on the edges are all one (i.e., G is the star graph with three leafs). We claim that G can not be embedded into Euclidean space with distortion $\leq \sqrt{2}$. Indeed, consider the associated metric space $(\mathsf{V}, \mathsf{d}_G)$ and an (expansive) embedding $f : \mathsf{V} \to \mathbb{R}^d$.



Consider the triangle formed by $\triangle = a'b'c'$, where $a' = f(a), b' = f(b)$ and $c' = f(c)$. Next, consider the following quantity $\max(\|a' - s'\|, \|b' - s'\|, \|c' - s'\|)$ which lower bounds the distortion of f. This quantity is minimized when $r = \|a' - s'\| = \|b' - s'\| = \|c' - s'\|$. Namely, s' is the center of the smallest enclosing circle of $\triangle$. However, r is minimized when all the edges of $\triangle$ are of equal length, and are of length $\mathsf{d}_G(a, b) = 2$. It follows that $\operatorname{dist}(f) \geq r \geq 2/\sqrt{3}$.

This quantity is minimized when $r = \|a' - s'\| = \|b' - s'\| = \|c' - s'\|$. Namely, s' is the center of the smallest enclosing circle of $\triangle$. However, r is minimized when all the edges of $\triangle$ are of equal length and are of length $\mathsf{d}_{\mathsf{G}}(a, b) = 2$. Observe that the height of the equilateral triangle with sidelength 2 is $h = \sqrt{3}$, and the radius of its inscribing circle is $r = (2/3)h = 2/\sqrt{3}$; see the figure on the right. As such, it follows that $\operatorname{dist}(f) \geq r = 2/\sqrt{3}$.



Note that the above argument is independent of the target dimension d. A packing argument shows that embedding the star graph with n leaves into $\mathbb{R}^d$ requires distortion $\Omega\left(n^{1/d}\right)$; see Exercise **??**. It

is known that $\Omega(\log n)$ distortion is necessary in the worst case when embedding a graph into Euclidean space (this is shown using expanders). A proof of distortion $\Omega(\log n / \log\log n)$ is sketched in the bibliographical notes.

25.2.1. Hierarchical Tree Metrics

The following metric is quite useful in practice, and nicely demonstrate why algorithmically finite metric spaces are useful.

Definition 25.2.1. ***Hierarchically well-separated tree*** (HST) is a metric space defined on the leaves of a rooted tree T. To each vertex $u \in T$ there is associated a label $\Delta_u \geq 0$ such that $\Delta_u = 0$ if and only if u is a leaf of T. The labels are such that if a vertex u is a child of a vertex v then $\Delta_u \leq \Delta_v$. The distance between two leaves $x, y \in T$ is defined as $\Delta_{\mathrm{lca}(x,y)}$, where $\mathrm{lca}(x, y)$ is the least common ancestor of x and y in T.

A HST T is a ***k-HST*** if for a vertex $v \in T$, we have that $\Delta_v \leq \Delta_{\overline{\mathrm{p}}(v)}/k$, where $\overline{\mathrm{p}}(v)$ is the parent of v in T.

Note that a HST is a very limited metric. For example, consider the cycle $G = C_n$ of n vertices, with weight one on the edges, and consider an expansive embedding f of G into a HST HST. It is easy to verify, that there must be two consecutive nodes of the cycle, which are mapped to two different subtrees of the root r of HST. Since HST is expansive, it follows that $\Delta_r \geq n/2$. As such, $\mathrm{dist}(f) \geq n/2$. Namely, HSTs fail to faithfully represent even very simple metrics.

25.2.2. Clustering

One natural problem we might want to solve on a graph (i.e., finite metric space) $(\mathcal{X}, \mathsf{d})$ is to partition it into clusters. One such natural clustering is the ***k-median clustering***, where we would like to choose a set $C \subseteq \mathcal{X}$ of k centers, such that $\nu_C(\mathcal{X}, \mathsf{d}) = \sum_{\mathsf{q} \in \mathcal{X}} \mathsf{d}(\mathsf{q}, C)$ is minimized, where $\mathsf{d}(\mathsf{q}, C) = \min_{c \in C} \mathsf{d}(\mathsf{q}, c)$ is the distance of q to its closest center in C.

It is known that finding the optimal k-median clustering in a (general weighted) graph is NP-complete. As such, the best we can hope for is an approximation algorithm. However, if the structure of the finite metric space $(\mathcal{X}, \mathsf{d})$ is simple, then the problem can be solved efficiently. For example, if the points of $\mathcal{X}$ are on the real line (and the distance between a and b is just $|a - b|$), then k-median can be solved using dynamic programming.

Another interesting case is when the metric space $(\mathcal{X}, \mathsf{d})$ is a HST. Is not too hard to prove the following lemma. See Exercise 25.7.1.

Lemma 25.2.2. *Let $(\mathcal{X}, \mathsf{d})$ be a HST defined over n points, and let $k > 0$ be an integer. One can compute the optimal k-median clustering of $\mathcal{X}$ in $O(k^2 n)$ time.*

Thus, if we can embed a general graph G into a HST HST, with low distortion, then we could approximate the k-median clustering on G by clustering the resulting HST, and "importing" the resulting partition to the original space. The quality of approximation, would be bounded by the distortion of the embedding of G into HST.

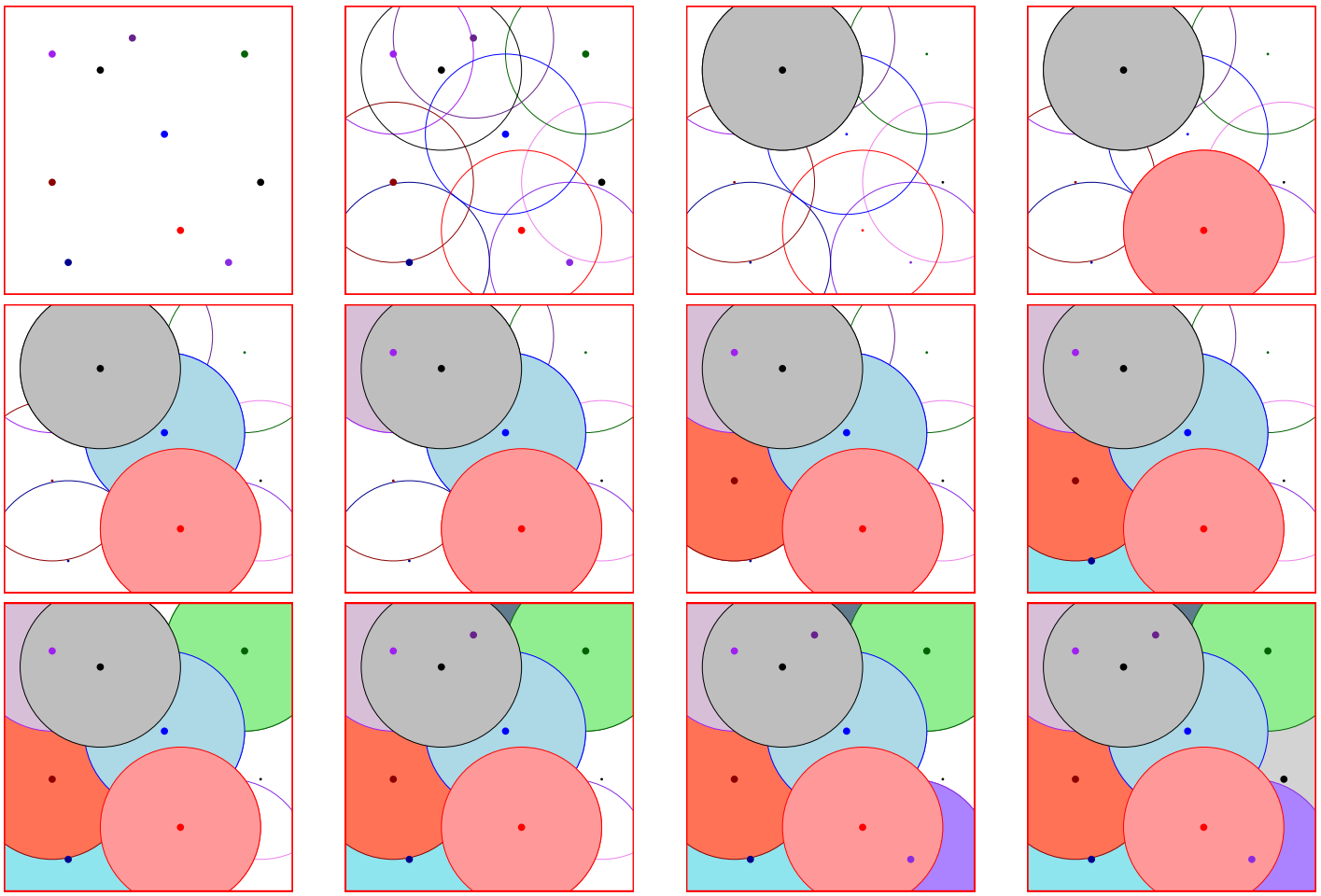


Figure 25.1: An example of the partition of a square (induced by a set of points) as described in Section 25.3.1.

25.3. Random Partitions

Let $(\mathcal{X}, d)$ be a finite metric space. Given a partition $P = \{C_1, \ldots, C_m\}$ of $\mathcal{X}$, we refer to the sets C_i as ***clusters***. We write $\mathcal{P}_{\mathcal{X}}$ for the set of all partitions of $\mathcal{X}$. For $x \in \mathcal{X}$ and a partition $P \in \mathcal{P}_{\mathcal{X}}$ we denote by $P(x)$ the unique cluster of P containing x. Finally, the set of all probability distributions on $\mathcal{P}_{\mathcal{X}}$ is denoted $\mathcal{D}_{\mathcal{X}}$.

The following partition scheme is due to [?].

25.3.1. Constructing the partition

Consider a given metric space $(\mathcal{X}, \mathsf{d})$, where $\mathcal{X}$ is a set of n points.

Let $\Delta = 2^u$ be a prescribed parameter, which is the required diameter of the resulting clusters. Choose, uniformly at random, a permutation π of $\mathcal{X}$ and a random value $\alpha \in [1/4, 1/2]$. Let $R = \alpha\Delta$, and observe that it is uniformly distributed in the interval $[\Delta/4, \Delta/2]$.

The partition is now defined as follows: A point $x \in \mathcal{X}$ is assigned to the cluster C_y of y, where y is the first point in the permutation in distance $\leq R$ from x. Formally,

$$C_y = \left\{ x \in \mathcal{X} \;\middle|\; x \in \mathbf{b}(y, R) \text{ and } \pi(y) \leq \pi(z) \text{ for all } z \in \mathcal{X} \text{ with } x \in \mathbf{b}(z, R) \right\}.$$

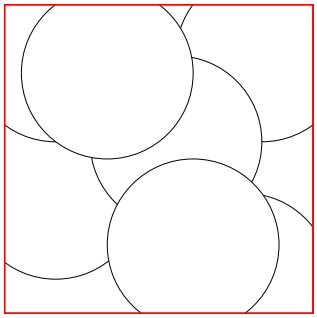


Figure 25.2: The resulting partition.

Let $P = \{C_y\}_{y \in \mathcal{X}}$ denote the resulting partition.

Here is a somewhat more intuitive explanation: Once we fix the radius of the clusters R, we start scooping out balls of radius R centered at the points of the random permutation π. At the ith stage, we scoop out only the remaining mass at the ball centered at x_i of radius r, where x_i is the ith point in the random permutation.

25.3.2. Properties

The following lemma quantifies the probability of a (crystal) ball of radius t centered at a point x is fully contained in one of the clusters of the partition? (Otherwise, the crystal ball is of course broken.)

Lemma 25.3.1. *Let $(\mathcal{X}, d)$ be a finite metric space, $\Delta = 2^u$ a prescribed parameter, and let P be the partition of $\mathcal{X}$ generated by the above random partition. Then the following holds:*

(i) For any $C \in P$, we have $\operatorname{diam}(C) \leq \Delta$.

(ii) Let x be any point of $\mathcal{X}$, and t a parameter $\leq \Delta/8$. Then,

$$\mathbb{P}\big[\mathbf{b}(x,t) \not\subseteq P(x)\big] \leq \frac{8t}{\Delta} \ln \frac{b}{a},$$

where $a = |\mathbf{b}(x, \Delta/8)|$, and $b = |\mathbf{b}(x, \Delta)|$.

Proof: Since $C_y \subseteq \mathbf{b}(y, R)$, we have that $\operatorname{diam}(C_y) \leq \Delta$, and thus the first claim holds.

Let U be the set of points of $\mathbf{b}(x, \Delta)$, such that $w \in U$ iff $\mathbf{b}(w, R) \cap \mathbf{b}(x, t) \neq \emptyset$. Arrange the points of U in increasing distance from x, and let $w_1, \ldots, w_{b'}$ denote the resulting order, where $b' = |U|$. Let $I_k = [d(x, w_k) - t, d(x, w_k) + t]$ and write $\mathcal{E}_k$ for the event that w_k is the first point in π such that $\mathbf{b}(x,t) \cap C_{w_k} \neq \emptyset$, and yet $\mathbf{b}(x,t) \not\subseteq C_{w_k}$. Note that if $w_k \in \mathbf{b}(x, \Delta/8)$, then $\mathbb{P}[\mathcal{E}_k] = 0$ since $\mathbf{b}(x,t) \subseteq \mathbf{b}(x, \Delta/8) \subseteq \mathbf{b}(w_k, \Delta/4) \subseteq \mathbf{b}(w_k, R)$.

In particular, $w_1, \ldots, w_a \in \mathbf{b}(x, \Delta/8)$ and as such $\mathbb{P}[\mathcal{E}_1] = \cdots = \mathbb{P}[\mathcal{E}_a] = 0$. Also, note that if $\mathsf{d}(x, w_k) < R - t$ then $\mathbf{b}(w_k, R)$ contains $\mathbf{b}(x,t)$ and as such $\mathcal{E}_k$ can not happen. Similarly, if $\mathsf{d}(x, w_k) > R + t$ then $\mathbf{b}(w_k, R) \cap \mathbf{b}(x,t) = \emptyset$ and $\mathcal{E}_k$ can not happen. As such, if $\mathcal{E}_k$ happen then $R - t \leq \mathsf{d}(x, w_k) \leq R + t$. Namely, if $\mathcal{E}_k$ happen then $R \in I_k$. Namely, $\mathbb{P}[\mathcal{E}_k] = \mathbb{P}[\mathcal{E}_k \cap (R \in I_k)] = \mathbb{P}[R \in I_k] \cdot \mathbb{P}[\mathcal{E}_k \mid R \in I_k]$. Now, R is uniformly distributed in the interval $[\Delta/4, \Delta/2]$, and I_k is an interval of length $2t$. Thus, $\mathbb{P}[R \in I_k] \leq 2t/(\Delta/4) = 8t/\Delta$.

Next, to bound $\mathbb{P}[\mathcal{E}_k \mid R \in I_k]$, we observe that $w_1, \ldots, w_{k-1}$ are closer to x than w_k and their distance to $\mathbf{b}(x,t)$ is smaller than R. Thus, if any of them appear before w_k in π then $\mathcal{E}_k$ does not happen. Thus, $\mathbb{P}[\mathcal{E}_k \mid R \in I_k]$ is bounded by the probability that w_k is the first to appear in π out of $w_1, \ldots, w_k$. But this probability is $1/k$, and thus $\mathbb{P}[\mathcal{E}_k \mid R \in I_k] \leq 1/k$.

We are now ready for the kill. Indeed,

$$\mathbb{P}[\mathbf{b}(x,t) \not\subseteq P(x)] = \sum_{k=1}^{b'} \mathbb{P}[\mathcal{E}_k] = \sum_{k=a+1}^{b'} \mathbb{P}[\mathcal{E}_k] = \sum_{k=a+1}^{b'} \mathbb{P}[R \in I_k] \cdot \mathbb{P}[\mathcal{E}_k \mid R \in I_k]$$
$$\leq \sum_{k=a+1}^{b'} \frac{8t}{\Delta} \cdot \frac{1}{k} \leq \frac{8t}{\Delta} \ln \frac{b'}{a} \leq \frac{8t}{\Delta} \ln \frac{b}{a},$$

since $\sum_{k=a+1}^{b} \frac{1}{k} \leq \int_a^b \frac{dx}{x} = \ln \frac{b}{a}$ and $b' \leq b$. ∎

25.4. Probabilistic embedding into trees

In this section, given n-point finite metric $(\mathcal{X}, \mathsf{d})$. we would like to embed it into a HST. As mentioned above, one can verify that for any embedding into HST, the distortion in the worst case is $\Omega(n)$. Thus, we define a randomized algorithm that embed $(\mathcal{X}, d)$ into a tree. Let T be the resulting tree, and consider two points $x, y \in \mathcal{X}$. Consider the ***random variable*** $\mathsf{d}_T(x,y)$. We constructed the tree T such that distances never shrink; i.e. $\mathsf{d}(x,y) \leq \mathsf{d}_T(x,y)$. The ***probabilistic distortion*** of this embedding is $\max_{x,y} \mathbb{E}\left[\frac{\mathsf{d}_T(x,y)}{\mathsf{d}(x,y)}\right]$. Somewhat surprisingly, one can find such an embedding with logarithmic probabilistic distortion.

Theorem 25.4.1. *Given n-point metric $(\mathcal{X}, d)$ one can randomly embed it into a 2-HST with probabilistic distortion $\leq 24 \ln n$.*

Proof: The construction is recursive. Let $\operatorname{diam}(P)$, and compute a random partition of $\mathcal{X}$ with cluster diameter $\operatorname{diam}(P)/2$, using the construction of Section 25.3.1. We recursively construct a 2-HST for each cluster, and hang the resulting clusters on the root node v, which is marked by $\Delta_v = \operatorname{diam}(P)$. Clearly, the resulting tree is a 2-HST.

For a node $v \in T$, let $\mathcal{X}(v)$ be the set of points of $\mathcal{X}$ contained in the subtree of v.

For the analysis, assume $\operatorname{diam}(P) = 1$, and consider two points $x, y \in \mathcal{X}$. We consider a node $v \in T$ to be in level i if $\operatorname{level}(v) = \lceil \lg \Delta_v \rceil = i$. The two points x and y correspond to two leaves in T, and let $\widehat{u}$ be the least common ancestor of x and y in t. We have $\mathsf{d}_T(x,y) \leq 2^{\operatorname{level}(v)}$. Furthermore, note that along a path the levels are strictly monotonically increasing.

Being more conservative, let w be the first ancestor of x, such that $\mathbf{b} = \mathbf{b}\big(x, \mathsf{d}(x,y)\big)$ is not completely contained in $\mathcal{X}(u_1), \ldots, \mathcal{X}(u_m)$, where $u_1, \ldots, u_m$ are the children of w. Clearly, $\operatorname{level}(w) > \operatorname{level}(\widehat{u})$. Thus, $\mathsf{d}_T(x,y) \leq 2^{\operatorname{level}(w)}$.

Consider the path σ from the root of T to x, and let $\mathcal{E}_i$ be the event that $\mathbf{b}$ is not fully contained in $\mathcal{X}(v_i)$, where v_i is the node of σ of level i (if such a node exists). Furthermore, let Y_i be the indicator variable which is 1 if $\mathcal{E}_i$ is the first to happened out of the sequence of events $\mathcal{E}_0, \mathcal{E}_{-1}, \ldots$. Clearly, $\mathsf{d}_T(x,y) \leq \sum Y_i 2^i$.

Let $t = \mathsf{d}(x,y)$ and $j = \lfloor \lg \mathsf{d}(x,y) \rfloor$, and $n_i = \left|\mathbf{b}(x, 2^i)\right|$ for $i = 0, \ldots, -\infty$. We have

$$\mathbb{E}[\mathsf{d}_T(x,y)] \leq \sum_{i=j}^{0} \mathbb{E}[Y_i]\, 2^i \leq \sum_{i=j}^{0} 2^i\, \mathbb{P}\left[\mathcal{E}_i \cap \overline{\mathcal{E}_{i-1}} \cap \overline{\mathcal{E}_{i-1}} \cdots \overline{\mathcal{E}_0}\right] \leq \sum_{i=j}^{0} 2^i \cdot \frac{8t}{2^i} \ln \frac{n_i}{n_{i-3}},$$

by Lemma 25.3.1. Thus,

$$\mathbb{E}[\mathsf{d}_T(x,y)] \leq 8t \ln\left(\prod_{i=j}^{0} \frac{n_i}{n_{i-3}}\right) \leq 8t \ln(n_0 \cdot n_1 \cdot n_2) \leq 24t \ln n.$$

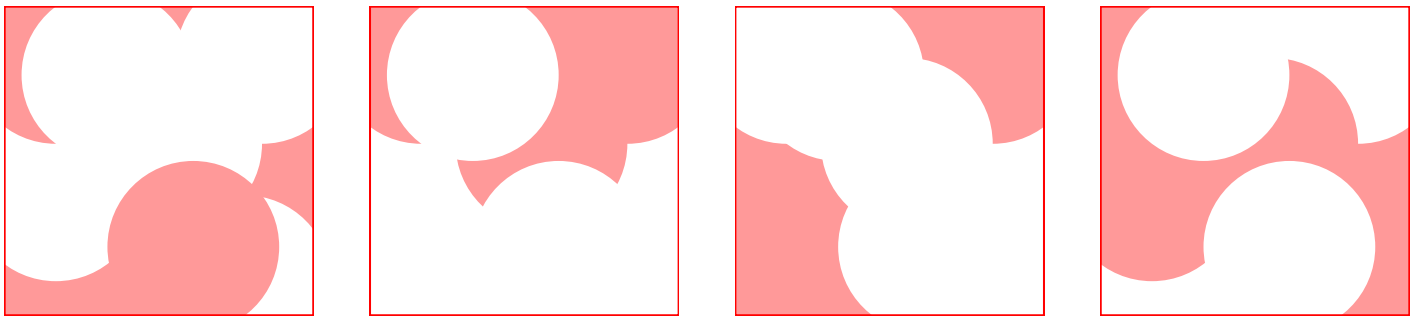


Figure 25.3: Examples of the sets resulting from the partition of Figure 25.1 and taking clusters into a set with probability 1/2.

It thus follows, that the expected distortion for x and y is $\leq 24 \ln n$. ∎

25.4.1. Application: approximation algorithm for k-median clustering

Let $(\mathcal{X}, \mathsf{d})$ be a n-point metric space, and let k be an integer number. We would like to compute the optimal k-median clustering. Number, find a subset $C_{\mathrm{opt}} \subseteq \mathcal{X}$, such that $\nu_{C_{\mathrm{opt}}}(\mathcal{X}, \mathsf{d})$ is minimized, see Section 25.2.2. To this end, we randomly embed $(\mathcal{X}, \mathsf{d})$ into a HST HST using Theorem 25.4.1. Next, using Lemma 25.2.2, we compute the optimal k-median clustering of HST. Let C be the set of centers computed. We return C together with the partition of $\mathcal{X}$ it induces as the required clustering.

Theorem 25.4.2. *Let $(\mathcal{X}, \mathsf{d})$ be a n-point metric space. One can compute in polynomial time a k-median clustering of $\mathcal{X}$ which has expected price $O(\alpha \log n)$, where α is the price of the optimal k-median clustering of $(\mathcal{X}, \mathsf{d})$.*

Proof: The algorithm is described above, and the fact that its running time is polynomial can be easily be verified. To prove the bound on the quality of the clustering, for any point $p \in \mathcal{X}$, let $\mathrm{cen}(p)$ denote the closest point in C_{opt} to p according to d, where C_{opt} is the set of k-medians in the optimal clustering. Let C be the set of k-medians returned by the algorithm, and let HST be the HST used by the algorithm. We have

$$\beta = \nu_C(\mathcal{X}, \mathsf{d}) \leq \nu_C(\mathcal{X}, \mathsf{d}_{\mathsf{HST}}) \leq \nu_{C_{\mathrm{opt}}}(\mathcal{X}, \mathsf{d}_{\mathsf{HST}}) \leq \sum_{p \in \mathcal{X}} \mathsf{d}_{\mathsf{HST}}(p, C_{\mathrm{opt}}) \leq \sum_{p \in \mathcal{X}} \mathsf{d}_{\mathsf{HST}}(p, \mathrm{cen}(p)).$$

Thus, in expectation we have

$$\begin{aligned}\mathbb{E}[\beta] &= \mathbb{E}\Big[\sum_{p \in \mathcal{X}} \mathsf{d}_{\mathsf{HST}}(p, \mathrm{cen}(p))\Big] = \sum_{p \in \mathcal{X}} \mathbb{E}[\mathsf{d}_{\mathsf{HST}}(p, \mathrm{cen}(p))] = \sum_{p \in \mathcal{X}} O(\mathsf{d}(p, \mathrm{cen}(p)) \log n) \\ &= O\Big((\log n) \sum_{p \in \mathcal{X}} \mathsf{d}(p, \mathrm{cen}(p))\Big) = O\Big(\nu_{C_{\mathrm{opt}}}(\mathcal{X}, \mathsf{d}) \log n\Big),\end{aligned}$$

by linearity of expectation and Theorem 25.4.1. ∎

25.5. Embedding any metric space into Euclidean space

Lemma 25.5.1. *Let $(\mathcal{X}, \mathsf{d})$ be a metric, and let $Y \subset \mathcal{X}$. Consider the mapping $f : \mathcal{X} \to \mathbb{R}$, where $f(x) = \mathsf{d}(x, Y) = \min_{y \in Y} \mathsf{d}(x, y)$. Then for any $x, y \in \mathcal{X}$, we have $|f(x) - f(y)| \leq \mathsf{d}(x, y)$. Namely f is nonexpansive.*

Proof: Indeed, let x' and y' be the closet points of Y, to x and y, respectively. Observe that

$$f(x) = \mathsf{d}(x, x') \leq \mathsf{d}(x, y') \leq \mathsf{d}(x, y) + \mathsf{d}(y, y') = \mathsf{d}(x, y) + f(y)$$

by the triangle inequality. Thus, $f(x) - f(y) \leq \mathsf{d}(x, y)$. By symmetry, we have $f(y) - f(x) \leq \mathsf{d}(x, y)$. Thus, $|f(x) - f(y)| \leq \mathsf{d}(x, y)$. ∎

25.5.1. The bounded spread case

Let $(\mathcal{X}, \mathsf{d})$ be a n-point metric. The ***spread*** of $\mathcal{X}$, denoted by

$$\Phi(\mathcal{X}) = \frac{\operatorname{diam}(\mathcal{X})}{\min_{x,y \in \mathcal{X}, x \neq y} \mathsf{d}(x, y)},$$

is the ratio between the diameter of $\mathcal{X}$ and the distance between the closest pair of points.

Theorem 25.5.2. *Given a n-point metric $\mathcal{Y} = (\mathcal{X}, d)$, with spread Φ, one can embed it into Euclidean space $\mathbb{R}^k$ with distortion $O\big(\sqrt{\ln \Phi} \ln n\big)$, where $k = O(\ln \Phi \ln n)$.*

Proof: Assume that $\operatorname{diam}(\mathcal{Y}) = \Phi$ (i.e., the smallest distance in $\mathcal{Y}$ is 1), and let $r_i = 2^{i-2}$, for $i = 1, \ldots, \alpha$, where $\alpha = \lceil \lg \Phi \rceil$. Let $P_{i,j}$ be a random partition of P with diameter r_i, using Theorem 25.4.1, for $i = 1, \ldots, \alpha$ and $j = 1, \ldots, \beta$, where $\beta = \lceil c \log n \rceil$ and c is a large enough constant to be determined shortly.

For each cluster of $P_{i,j}$ randomly toss a coin, and let $V_{i,j}$ be the all the points of $\mathcal{X}$ that belong to clusters in $P_{i,j}$ that got 'T' in their coin toss. For a point $u \in x$, let

$$f_{i,j}(x) = \mathsf{d}(x, \mathcal{X} \setminus V_{i,j}) = \min_{v \in \mathcal{X} \setminus V_{i,j}} \mathsf{d}(x, v),$$

for $i = 0, \ldots, m$ and $j = 1, \ldots, \beta$. Let $F : \mathcal{X} \to \mathbb{R}^{(m+1)\cdot\beta}$ be the embedding, such that

$$F(x) = \big(\underbrace{f_{0,1}(x), f_{0,2}(x), \ldots, f_{0,\beta}(x)}_{\text{first n resolution block}}, f_{1,1}(x), f_{1,2}(x), \ldots, f_{1,\beta}(x), \ldots, f_{m,1}(x), f_{m,2}(x), \ldots, f_{\alpha,\beta}(x)\big).$$

Next, consider two points $x, y \in \mathcal{X}$, with distance $\phi = \mathsf{d}(x, y)$. Let k be an integer such that $r_u \leq \phi/2 \leq r_{u+1}$. Clearly, in any partition of $P_{u,1}, \ldots, P_{u,\beta}$ the points x and y belong to different clusters. Furthermore, with probability half $x \in V_{u,j}$ and $y \notin V_{u,j}$ or $x \notin V_{u,j}$ and $y \in V_{u,j}$, for $1 \leq j \leq \beta$.

Let $\mathcal{E}_j$ denote the event that $\mathbf{b}(x, \rho) \subseteq V_{u,j}$ and $y \notin V_{u,j}$, for $j = 1, \ldots, \beta$, where $\rho = \phi/(64 \ln n)$. By Lemma 25.3.1, we have

$$\mathbb{P}\big[\mathbf{b}(x, \rho) \not\subseteq P_{u,j}(x)\big] \leq \frac{8\rho}{r_u} \ln n \leq \frac{\phi}{8 r_u} \leq 1/2.$$

Thus,

$$\begin{aligned}
\mathbb{P}\big[\mathcal{E}_j\big] &= \mathbb{P}\big[\big(\mathbf{b}(x, \rho) \subseteq P_{u,j}(x)\big) \cap \big(x \in V_{u,j}\big) \cap \big(y \notin V_{u,j}\big)\big] \\
&= \mathbb{P}\big[\mathbf{b}(x, \rho) \subseteq P_{u,j}(x)\big] \cdot \mathbb{P}\big[x \in V_{u,j}\big] \cdot \mathbb{P}\big[y \notin V_{u,j}\big] \geq 1/8,
\end{aligned}$$

since those three events are independent. Notice, that if $\mathcal{E}_j$ happens, than $f_{u,j}(x) \geq \rho$ and $f_{u,j}(y) = 0$.

Let X_j be an indicator variable which is 1 if $\mathcal{E}_j$ happens, for $j = 1, \ldots, \beta$. Let $Z = \sum_j X_j$, and we have $\mu = \mathbb{E}[Z] = \mathbb{E}\big[\sum_j X_j\big] \geq \beta/8$. Thus, the probability that only $\beta/16$ of $\mathcal{E}_1, \ldots, \mathcal{E}_\beta$ happens, is

$\mathbb{P}[Z < (1-1/2)\,\mathbb{E}[Z]]$. By the Chernoff inequality, we have $\mathbb{P}[Z < (1-1/2)\,\mathbb{E}[Z]] \le \exp\bigl(-\mu 1/(2\cdot 2^2)\bigr) = \exp(-\beta/64) \le 1/n^{10}$, if we set $c = 640$.

Thus, with high probability

$$\|F(x) - F(y)\| \ge \sqrt{\sum_{j=1}^{\beta} \bigl(f_{u,j}(x) - f_{u,j}(y)\bigr)^2} \ge \sqrt{\rho^2 \frac{\beta}{16}} = \sqrt{\beta}\frac{\rho}{4} = \phi \cdot \frac{\sqrt{\beta}}{256 \ln n}.$$

On the other hand, $\bigl|f_{i,j}(x) - f_{i,j}(y)\bigr| \le \mathsf{d}(x,y) = \phi \le 64\rho \ln n$. Thus,

$$\|F(x) - F(y)\| \le \sqrt{\alpha\beta(64\rho \ln n)^2} \le 64\sqrt{\alpha\beta}\rho \ln n = \sqrt{\alpha\beta}\cdot\phi.$$

Thus, setting $G(x) = F(x)\frac{256 \ln n}{\sqrt{\beta}}$, we get a mapping that maps two points of distance ϕ from each other to two points with distance in the range $\left[\phi, \phi\cdot\sqrt{\alpha\beta}\cdot\frac{256\ln n}{\sqrt{\beta}}\right]$. Namely, $G(\cdot)$ is an embedding with distortion $O(\sqrt{\alpha}\ln n) = O(\sqrt{\ln \Phi}\ln n)$.

The probability that G fails on one of the pairs, is smaller than $(1/n^{10})\cdot\binom{n}{2} < 1/n^8$. In particular, we can check the distortion of G for all $\binom{n}{2}$ pairs, and if any of them fail (i.e., the distortion is too big), we restart the process. ∎

25.5.2. The unbounded spread case

Our next task, is to extend Theorem 25.5.2 to the case of unbounded spread. Indeed, let $(\mathcal{X}, d)$ be a n-point metric, such that $\operatorname{diam}(\mathcal{X}) \le 1/2$. Again, we look on the different resolutions $r_1, r_2, \ldots$, where $r_i = 1/2^{i-1}$. For each one of those resolutions r_i, we can embed this resolution into β coordinates, as done for the bounded case. Then we concatenate the coordinates together.

There are two problems with this approach: (i) the number of resulting coordinates is infinite, and (ii) a pair x, y, might be distorted a "lot" because it contributes to all resolutions, not only to its "relevant" resolutions.

Both problems can be overcome with careful tinkering. Indeed, for a resolution r_i, we are going to modify the metric, so that it ignores short distances (i.e., distances $\le r_i/n^2$). Formally, for each resolution r_i, let $G_i = (\mathcal{X}, \widehat{E}_i)$ be the graph where two points x and y are connected if $\mathsf{d}(x,y) \le r_i/n^2$. Consider a connected component $C \in G_i$. For any two points $x, y \in C$, we have $\mathsf{d}(x,y) \le n(r_i/n^2) \le r_i/n$. Let $\mathcal{X}_i$ be the set of connected components of G_i, and define the distances between two connected components $C, C' \in \mathcal{X}_i$, to be $\mathsf{d}_i(C,C') = \mathsf{d}(C,C') = \min_{c\in C, c'\in C'} \mathsf{d}(c,c')$.

It is easy to verify that $(\mathcal{X}_i, \mathsf{d}_i)$ is a metric space (see Exercise 25.7.2). Furthermore, we can naturally embed $(\mathcal{X}, \mathsf{d})$ into $(\mathcal{X}_i, \mathsf{d}_i)$ by mapping a point $x \in \mathcal{X}$ to its connected components in $\mathcal{X}_i$. Essentially $(\mathcal{X}_i, \mathsf{d}_i)$ is a snapped version of the metric $(\mathcal{X}, d)$, with the advantage that $\Phi((\mathcal{X}, \mathsf{d}_i)) = O(n^2)$. We now embed $\mathcal{X}_i$ into $\beta = O(\log n)$ coordinates. Next, for any point of $\mathcal{X}$ we embed it into those β coordinates, by using the embedding of its connected component in $\mathcal{X}_i$. Let E_i be the embedding for resolution r_i. Namely, $E_i(x) = (f_{i,1}(x), f_{i,2}(x), \ldots, f_{i,\beta}(x))$, where $f_{i,j}(x) = \min(\mathsf{d}_i(x, \mathcal{X}\setminus V_{i,j}), 2r_i)$. The resulting embedding is $F(x) = \oplus E_i(x) = (E_1(x), E_2(x), \ldots,)$.

Since we slightly modified the definition of $f_{i,j}(\cdot)$, we have to show that $f_{i,j}(\cdot)$ is nonexpansive. Indeed, consider two points $x, y \in \mathcal{X}_i$, and observe that

$$\bigl|f_{i,j}(x) - f_{i,j}(y)\bigr| \le \bigl|\mathsf{d}_i(x, V_{i,j}) - \mathsf{d}_i(y, V_{i,j})\bigr| \le \mathsf{d}_i(x,y) \le \mathsf{d}(x,y),$$

as a simple case analysis[①] shows.

For a pair $x, y \in \mathcal{X}$, and let $\phi = \mathsf{d}(x, y)$. To see that $F(\cdot)$ is the required embedding (up to scaling), observe that, by the same argumentation of Theorem 25.5.2, we have that with high probability

$$\|F(x) - F(y)\| \geq \phi \cdot \frac{\sqrt{\beta}}{256 \ln n}.$$

To get an upper bound on this distance, observe that for i such that $r_i > \phi n^2$, we have $E_i(x) = E_i(y)$. Thus,

$$\begin{aligned}
\|F(x) - F(y)\|^2 &= \sum_i \|E_i(x) - E_i(y)\|^2 = \sum_{i, r_i < \phi n^2} \|E_i(x) - E_i(y)\|^2 \\
&= \sum_{i, \phi/n^2 < r_i < \phi n^2} \|E_i(x) - E_i(y)\|^2 + \sum_{i, r_i < \phi/n^2} \|E_i(x) - E_i(y)\|^2 \\
&= \beta\phi^2 \lg\left(n^4\right) + \sum_{i, r_i < \phi/n^2} (2r_i)^2 \beta \leq 4\beta\phi^2 \lg n + \frac{4\phi^2\beta}{n^4} \leq 5\beta\phi^2 \lg n.
\end{aligned}$$

Thus, $\|F(x) - F(y)\| \leq \phi\sqrt{5\beta \lg n}$. We conclude, that with high probability, $F(\cdot)$ is an embedding of $\mathcal{X}$ into Euclidean space with distortion $\left(\phi\sqrt{5\beta \lg n}\right) / \left(\phi \cdot \frac{\sqrt{\beta}}{256 \ln n}\right) = O(\log^{3/2} n)$.

We still have to handle the infinite number of coordinates problem. However, the above proof shows that we care about a resolution r_i (i.e., it contributes to the estimates in the above proof) only if there is a pair x and y such that $r_i/n^2 \leq \mathsf{d}(x, y) \leq r_i n^2$. Thus, for every pair of distances there are $O(\log n)$ relevant resolutions. Thus, there are at most $\eta = O(n^2 \beta \log n) = O(n^2 \log^2 n)$ relevant coordinates, and we can ignore all the other coordinates. Next, consider the affine subspace h that spans $F(P)$. Clearly, it is $n - 1$ dimensional, and consider the projection $G : \mathbb{R}^\eta \to \mathbb{R}^{n-1}$ that projects a point to its closest point in h. Clearly, $G(F(\cdot))$ is an embedding with the same distortion for P, and the target space is of dimension $n - 1$.

Note, that all this process succeeds with high probability. If it fails, we try again. We conclude:

Theorem 25.5.3 (Low quality Bourgain theorem). *Given a n-point metric M, one can embed it into Euclidean space of dimension $n-1$, such that the distortion of the embedding is at most $O(\log^{3/2} n)$.*

Using the Johnson-Lindenstrauss lemma, the dimension can be further reduced to $O(\log n)$. Being more careful in the proof, it is possible to reduce the dimension to $O(\log n)$ directly.

25.6. Bibliographical notes

The partitions we use are due to Calinescu *et al.* [**?**]. The idea of embedding into spanning trees is due to Alon *et al.* [AKPW95], which showed that one can get a probabilistic distortion of $2^{O\left(\sqrt{\log n \log\log n}\right)}$. Yair Bartal realized that by allowing trees with additional vertices, one can get a considerably better result. In particular, he showed [Bar96] that probabilistic embedding into trees can be done with polylogarithmic average distortion. He later improved the distortion to $O(\log n \log\log n)$ in [Bar98]. Improving this

[①] Indeed, if $f_{i,j}(x) < \mathsf{d}_i(x, V_{i,j})$ and $f_{i,j}(y) < \mathsf{d}_i(x, V_{i,j})$ then $f_{i,j}(x) = 2r_i$ and $f_{i,j}(y) = 2r_i$, which implies the above inequality. If $f_{i,j}(x) = \mathsf{d}_i(x, V_{i,j})$ and $f_{i,j}(y) = \mathsf{d}_i(x, V_{i,j})$ then the inequality trivially holds. The other option is handled in a similar fashion.

result was an open question, culminating in the work of Fakcharoenphol *et al.* [FRT04] which achieve the optimal $O(\log n)$ distortion.

Our proof of Lemma 25.3.1 (which is originally from [FRT04]) is taken from [KLMN05]. The proof of Theorem 25.5.3 is by Gupta [Gup00].

A good exposition of metric spaces is available in Matoušek [Mat02].

Embedding into spanning trees. The above embeds the graph into a Steiner tree. A more useful representation, would be a random embedding into a spanning tree. Surprisingly, this can be done, as shown by Emek *et al.* [EEST08]. This was improved to $O(\log n \cdot \log\log n \cdot (\log\log\log n)^3)$[②] by Abraham *et al.* [ABN08a, ABN08b].

Alternative proof of the tree embedding result. Interestingly, if one does not care about the optimal distortion, one can get similar result (for embedding into probabilistic trees), by first embedding the metric into Euclidean space, then reduce the dimension by the Johnson-Lindenstrauss lemma, and finally, construct an HST by constructing a quadtree over the points. The "trick" is to randomly translate the quadtree. It is easy to verify that this yields $O(\log^4 n)$ distortion. See the survey by Indyk [Ind01] for more details. This random shifting of quadtrees is a powerful technique that was used in getting several result, and it is a crucial ingredient in Arora [Aro98] approximation algorithm for Euclidean TSP.

25.7. Exercises

Exercise 25.7.1 (Clustering for HST). Let $(\mathcal{X}, \mathsf{d})$ be a HST defined over n points, and let $k > 0$ be an integer. Provide an algorithm that computes the optimal k-median clustering of $\mathcal{X}$ in $O(k^2 n)$ time.

[Transform the HST into a tree where every node has only two children. Next, run a dynamic programming algorithm on this tree.]

Exercise 25.7.2 (Partition induced metric).

(a) Give a counter example to the following claim: Let $(\mathcal{X}, \mathsf{d})$ be a metric space, and let P be a partition of $\mathcal{X}$. Then, the pair (P, d') is a metric, where $\mathsf{d}'(C, C') = \mathsf{d}(C, C') = \min_{x \in C, y \in C'} \mathsf{d}(x, y)$ and $C, C' \in P$.

(b) Let $(\mathcal{X}, \mathsf{d})$ be a n-point metric space, and consider the set $U = \left\{ i \;\middle|\; 2^i \le \mathsf{d}(x, y) \le 2^{i+1}, \text{ for } x, y \in \mathcal{X} \right\}$. Prove that $|U| = O(n)$. Namely, there are only n different resolutions that "matter" for a finite metric space.

Exercise 25.7.3 (Computing the diameter via embeddings).

(a) (h:1) Let ℓ be a line in the plane, and consider the embedding $f : \mathbb{R}^2 \to \ell$, which is the projection of the plane into ℓ. Prove that f is 1-Lipschitz, but it is not K-bi-Lipschitz for any constant K.

(b) (h:3) Prove that one can find a family of projections $\mathcal{F}$ of size $O(1/\sqrt{\varepsilon})$, such that for any two points $x, y \in \mathbb{R}^2$, for one of the projections $f \in \mathcal{F}$ we have $\mathsf{d}(f(x), f(y)) \ge (1 - \varepsilon)\mathsf{d}(x, y)$.

(c) (h:1) Given a set P of n in the plane, given a $O(n/\sqrt{\varepsilon})$ time algorithm that outputs two points $x, y \in P$, such that $\mathsf{d}(x, y) \ge (1 - \varepsilon)\mathrm{diam}(P)$, where $\mathrm{diam}(P) = \max_{z, w \in P} \mathsf{d}(z, w)$ is the diameter of P.

[②] Truely a polyglot of logs.

(d) (h:2) Given P, show how to extract, in $O(n)$ time, a set $Q \subseteq P$ of size $O(\varepsilon^{-2})$, such that $\mathrm{diam}(Q) \geq (1 - \varepsilon/2)\mathrm{diam}(P)$. (Hint: Construct a grid of appropriate resolution.)

In particular, give an $(1-\varepsilon)$-approximation algorithm to the diameter of P that works in $O(n+\varepsilon^{-2.5})$ time. (There are slightly faster approximation algorithms known for approximating the diameter.)

Acknowledgments

The presentation in this write-up follows closely the insightful suggestions of Manor Mendel.

Chapter 26

Entropy, Randomness, and Information

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
December 10, 2019

> "If only once - only once - no matter where, no matter before what audience - I could better the record of the great Rastelli and juggle with thirteen balls, instead of my usual twelve, I would feel that I had truly accomplished something for my country. But I am not getting any younger, and although I am still at the peak of my powers there are moments - why deny it? - when I begin to doubt - and there is a time limit on all of us."
>
> Romain Gary, The talent scout

26.1. The entropy function

Definition 26.1.1. The ***entropy*** in bits of a discrete random variable X is given by

$$\mathbb{H}(X) = -\sum_x \mathbb{P}[X = x] \lg \mathbb{P}[X = x],$$

where $\lg x$ is the logarithm base 2 of x. Equivalently, $\mathbb{H}(X) = \mathbb{E}\left[\lg \frac{1}{\mathbb{P}[X]}\right]$.

The ***binary entropy*** function $\mathbb{H}(p)$ for a random binary variable that is 1 with probability p, is

$$\mathbb{H}(p) = -p \lg p - (1-p)\lg(1-p).$$

We define $\mathbb{H}(0) = \mathbb{H}(1) = 0$.

The function $\mathbb{H}(p)$ is a concave symmetric around $1/2$ on the interval $[0, 1]$ and achieves its maximum at $1/2$. For a concrete example, consider $\mathbb{H}(3/4) \approx 0.8113$ and $\mathbb{H}(7/8) \approx 0.5436$. Namely, a coin that has $3/4$ probably to be heads have higher amount of "randomness" in it than a coin that has probability $7/8$ for heads.

Writing $\lg n = (\ln n)/\ln 2$, we have that

$$\mathbb{H}(p) = \frac{1}{\ln 2}\big(-p \ln p - (1-p)\ln(1-p)\big)$$

$$\text{and} \quad \mathbb{H}'(p) = \frac{1}{\ln 2}\left(-\ln p - \frac{p}{p} - (-1)\ln(1-p) - \frac{1-p}{1-p}(-1)\right) = \lg \frac{1-p}{p}.$$

Deploying our amazing ability to compute derivative of simple functions once more, we get that

$$\mathbb{H}''(p) = \frac{1}{\ln 2}\frac{p}{1-p}\left(\frac{p(-1) - (1-p)}{p^2}\right) = -\frac{1}{p(1-p)\ln 2}.$$

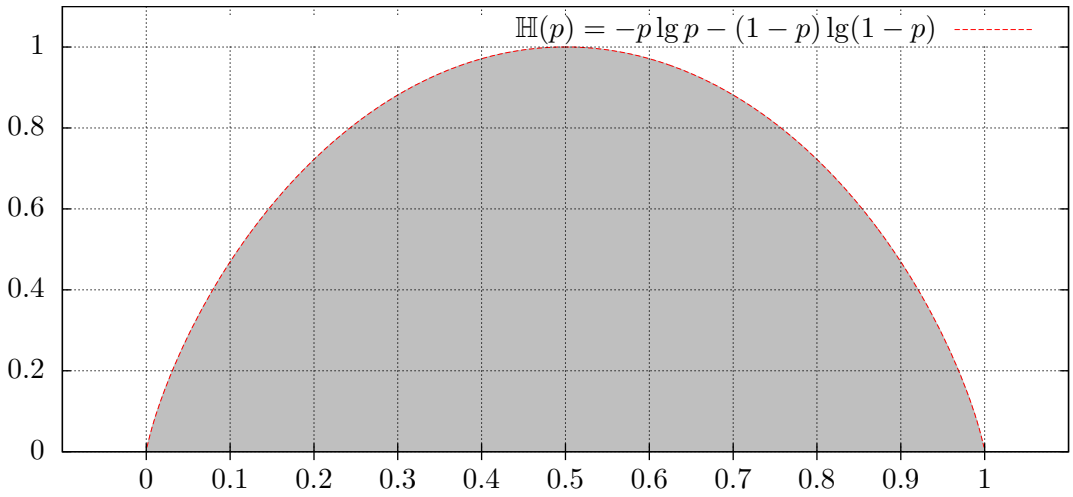


Figure 26.1: The binary entropy function.

Since $\ln 2 \approx 0.693$, we have that $\mathbb{H}''(p) \leq 0$, for all $p \in (0,1)$, and the $\mathbb{H}(\cdot)$ is concave in this range. Also, $\mathbb{H}'(1/2) = 0$, which implies that $\mathbb{H}(1/2) = 1$ is a maximum of the binary entropy. Namely, a balanced coin has the largest amount of randomness in it.

Example 26.1.2. A random variable X that has probability $1/n$ to be i, for $i = 1, \ldots, n$, has entropy $\mathbb{H}(X) = -\sum_{i=1}^{n} \frac{1}{n} \lg \frac{1}{n} = \lg n$.

Note, that the entropy is oblivious to the exact values that the random variable can have, and it is sensitive only to the probability distribution. Thus, a random variables that accepts $-1, +1$ with equal probability has the same entropy (i.e., 1) as a fair coin.

Lemma 26.1.3. *Let X and Y be two independent random variables, and let Z be the random variable (X, T). Then $\mathbb{H}(Z) = \mathbb{H}(X) + \mathbb{H}(Y)$.*

Proof: In the following, summation are over all possible values that the variables can have. By the independence of X and Y we have

$$\begin{aligned}
\mathbb{H}(Z) &= \sum_{x,y} \mathbb{P}[(X,Y) = (x,y)] \lg \frac{1}{\mathbb{P}[(X,Y) = (x,y)]} \\
&= \sum_{x,y} \mathbb{P}[X = x]\, \mathbb{P}[Y = y] \lg \frac{1}{\mathbb{P}[X = x]\, \mathbb{P}[Y = y]} \\
&= \sum_{x} \sum_{y} \mathbb{P}[X = x]\, \mathbb{P}[Y = y] \lg \frac{1}{\mathbb{P}[X = x]} \\
&\quad + \sum_{y} \sum_{x} \mathbb{P}[X = x]\, \mathbb{P}[Y = y] \lg \frac{1}{\mathbb{P}[Y = y]} \\
&= \sum_{x} \mathbb{P}[X = x] \lg \frac{1}{\mathbb{P}[X = x]} + \sum_{y} \mathbb{P}[Y = y] \lg \frac{1}{\mathbb{P}[Y = y]} = \mathbb{H}(X) + \mathbb{H}(Y).
\end{aligned}$$

■

Lemma 26.1.4. *Suppose that nq is integer in the range $[0, n]$. Then $\frac{2^{n\mathbb{H}(q)}}{n+1} \leq \binom{n}{nq} \leq 2^{n\mathbb{H}(q)}$.*

Proof: This trivially holds if $q = 0$ or $q = 1$, so assume $0 < q < 1$. We know that

$$\begin{aligned} & \binom{n}{nq} q^{nq}(1-q)^{n-nq} \le (q+(1-q))^n = 1 \\ \implies \qquad & \binom{n}{nq} \le q^{-nq}(1-q)^{-n(1-q)} = 2^{n\,(-q\lg q-(1-q)\lg(1-q))} = 2^{n\mathbb{H}(q)}. \end{aligned}$$

As for the other direction, let

$$\mu(k) = \binom{n}{k} q^k (1-q)^{n-k}.$$

The claim is that $\mu(nq)$ is the largest term in $\sum_{k=0}^{n} \mu(k) = 1$, where $\mu(k) = \binom{n}{k} q^k (1-q)^{n-k}$. Indeed,

$$\Delta_k = \mu(k) - \mu(k+1) = \binom{n}{k} q^k (1-q)^{n-k} \left(1 - \frac{n-k}{k+1}\frac{q}{1-q}\right),$$

and the sign of this quantity is the sign of $(k+1)(1-q)-(n-k)q = k+1-kq-q-nq+kq = 1+k-q-nq$. Namely, $\Delta_k \ge 0$ when $k \ge nq+q-1$, and $\Delta_k < 0$ otherwise. Namely, $\mu(k) < \mu(k+1)$, for $k < nq$, and $\mu(k) \ge \mu(k+1)$ for $k \ge nq$. Namely, $\mu(nq)$ is the largest term in $\sum_{k=0}^{n} \mu(k) = 1$, and as such it is larger than the average. We have $\mu(nq) = \binom{n}{nq} q^{nq}(1-q)^{n-nq} \ge \frac{1}{n+1}$, which implies

$$\binom{n}{nq} \ge \frac{1}{n+1} q^{-nq}(1-q)^{-(n-nq)} = \frac{1}{n+1} 2^{n\mathbb{H}(q)}.$$

■

Lemma 26.1.4 can be extended to handle non-integer values of q. This is straightforward, and we omit the easy details.

Corollary 26.1.5. *We have:*

(i) $q \in [0, 1/2] \Rightarrow \binom{n}{\lfloor nq \rfloor} \le 2^{n\mathbb{H}(q)}$.

(ii) $q \in [1/2, 1] \Rightarrow \binom{n}{\lceil nq \rceil} \le 2^{n\mathbb{H}(q)}$.

(iii) $q \in [1/2, 1] \Rightarrow \frac{2^{n\mathbb{H}(q)}}{n+1} \le \binom{n}{\lfloor nq \rfloor}$.

(iv) $q \in [0, 1/2] \Rightarrow \frac{2^{n\mathbb{H}(q)}}{n+1} \le \binom{n}{\lceil nq \rceil}$.

The bounds of Lemma 26.1.4 and Corollary 26.1.5 are loose but sufficient for our purposes. As a sanity check, consider the case when we generate a sequence of n bits using a coin with probability q for head, then by the Chernoff inequality, we will get roughly nq heads in this sequence. As such, the generated sequence Y belongs to $\binom{n}{nq} \approx 2^{n\mathbb{H}(q)}$ possible sequences that have similar probability. As such, $\mathbb{H}(Y) \approx \lg \binom{n}{nq} = n\mathbb{H}(q)$, by Example 26.1.2, this also readily follows from Lemma 26.1.3.

26.2. Extracting randomness

The problem. We are given a random variable X that is chosen uniformly at random from $[\![0 : m-1]\!] = \{0, \ldots, m-1\}$. Our purpose is built an algorithm that given X output a binary string, such that the bits in the binary string can be interpreted as the coin flips of a fair balanced coin. That is, the probability of the ith bit of the output (if it exists) to be 0 (or 1) is exactly half, and the different bits of the output are independent.

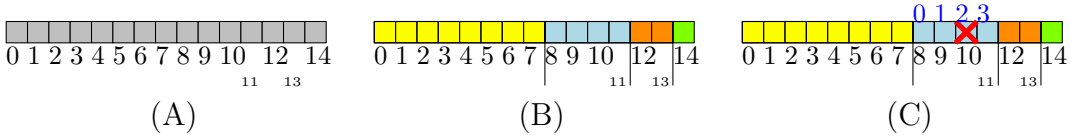


Figure 26.2: (A) $m = 15$. (B) The block decomposition. (C) If $X = 10$, then the extraction output is 2 in base 2, using 2 bits – that is 10.

Idea. We break the $[\![0 : m-1]\!]$ into consecutive blocks that are powers of two. Given the value of X, we find which block contains it, and we output a binary representation of the location of X in the block containing it, where if a block is length 2^k, then we output k bits.

Entropy can be interpreted as the amount of unbiased random coin flips can be extracted from a random variable.

Definition 26.2.1. An ***extraction function*** $\mathtt{Ext}$ takes as input the value of a random variable X and outputs a sequence of bits y, such that $\mathbb{P}\left[\mathtt{Ext}(X) = y \;\middle|\; |y| = k\right] = 1/2^k$. whenever $\mathbb{P}\left[|y| = k\right] \geq 0$, where $|y|$ denotes the length of y.

As a concrete (easy) example, consider X to be a uniform random integer variable out of $0, \ldots, 7$. All that $\mathtt{Ext}(x)$ has to do in this case, is just to compute the binary representation of x.

The definition of the extraction function has two subtleties:

(A) It requires that all extracted sequences of the same length (say k), have the same probability to be output (i.e., $1/2^k$).

(B) If the extraction function can output a sequence of length k, then it needs to be able to output *all* 2^k such binary sequences.

Thus, for X a uniform random integer variable in the range $0, \ldots, 11$, the function $\mathtt{Ext}(x)$ can output the binary representation for x if $0 \leq x \leq 7$. However, what do we do if x is between 8 and 11? The idea is to output the binary representation of $x - 8$ as a two bit number. Clearly, Definition 26.2.1 holds for this extraction function, since $\mathbb{P}\left[\mathtt{Ext}(X) = 00 \;\middle|\; |\mathtt{Ext}(X)| = 2\right] = 1/4$. as required. This scheme can be of course extracted for any range.

Tedium 26.2.2. For $x \leq y$ positive integers, and any positive integer Δ, we have that

$$\frac{x}{y} \leq \frac{x+\Delta}{y+\Delta} \iff x(y+\Delta) \leq y(x+\Delta) \iff x\Delta \leq y\Delta \iff x \leq y.$$

Theorem 26.2.3. *Suppose that the value of a random variable X is chosen uniformly at random from the integers $\{0, \ldots, m-1\}$. Then there is an extraction function for X that outputs on average (i.e., in expectation) at least $\lfloor \lg m \rfloor - 1 = \lfloor \mathbb{H}(X) \rfloor - 1$ independent and unbiased bits.*

Proof: We represent m as a sum of unique powers of 2, namely $m = \sum_i a_i 2^i$, where $a_i \in \{0, 1\}$. Thus, we decomposed $\{0, \ldots, m-1\}$ into a disjoint union of blocks that have sizes which are distinct powers of 2. If a number falls inside such a block, we output its relative location in the block, using binary representation of the appropriate length (i.e., k if the block is of size 2^k). It is not difficult to verify that this function fulfills the conditions of Definition 26.2.1, and it is thus an extraction function.

Now, observe that the claim holds if m is a power of two, by Example 26.1.2 (i.e., if $m = 2^k$, then $\mathbb{H}(X) = k$). Thus, if m is not a power of 2, then in the decomposition if there is a block of size 2^k, and the X falls inside this block, then the entropy is k.

The remainder of the proof is by induction – assume the claim holds if the range used by the random variable is strictly smaller than m. In particular, let $K = 2^k$ be the largest power of 2 that is smaller than m, and let $U = 2^u$ be the largest power of two such that $U \le m - K \le 2U$.

If the random number $X \in [\![0 : K-1]\!]$, then the scheme outputs k bits. Otherwise, we can think about the extraction function as being recursive and extracting randomness from a random variable $X' = X - K$ that is uniformly distributed in $[\![0 : m-K]\!]$.

By Tedium 26.2.2, we have that

$$\frac{m-K}{m} \le \frac{m-K+(2U+K-m)}{m+(2U+K-m)} = \frac{2U}{2U+K}$$

Let Y be the random variable which is the number of random bits extracted. We have that

$$\begin{aligned}\mathbb{E}[Y] &\ge \frac{K}{m}k + \frac{m-K}{m}(\lfloor \lg(m-K)\rfloor - 1) = k - \frac{m-K}{m}k + \frac{m-K}{m}(u-1) = k + \frac{m-K}{m}(\overbrace{u-k-1}^{<0}) \\ &\ge k - \frac{2U}{2U+K}(u-k-1) = k - \frac{2U}{2U+K}(1+k-u).\end{aligned}$$

If $u = k-1$, then $\mathbb{H}(X) \ge k - \frac{1}{2}\cdot 2 = k-1$, as required. If $u = k-2$ then $\mathbb{H}(X) \ge k - \frac{1}{3}\cdot 3 = k-1$. Finally, if $u < k-2$ then

$$\mathbb{E}[Y] \ge k - \frac{2U}{2U+K}(1+k-u) \ge k - \frac{2U}{K}(1+k-u) = k - \frac{k-u+1}{2^{(k-u+1)-2}} \ge k-1,$$

since $k-u+1 \ge 4$ and $i/2^{i-2} \le 1$ for $i \ge 4$. ∎

Theorem 26.2.4. *Consider a coin that comes up heads with probability $p > 1/2$. For any constant $\delta > 0$ and for n sufficiently large:*

(A) One can extract, from an input of a sequence of n flips, an output sequence of $(1-\delta)n\mathbb{H}(p)$ (unbiased) independent random bits.

(B) One can not extract more than $n\mathbb{H}(p)$ bits from such a sequence.

Proof: There are $\binom{n}{j}$ input sequences with exactly j heads, and each has probability $p^j(1-p)^{n-j}$. We map this sequence to the corresponding number in the set $\left\{0,\ldots,\binom{n}{j}-1\right\}$. Note, that this, conditional distribution on j, is uniform on this set, and we can apply the extraction algorithm of Theorem 26.2.3. Let Z be the random variables which is the number of heads in the input, and let B be the number of random bits extracted. We have

$$\mathbb{E}[B] = \sum_{k=0}^{n} \mathbb{P}[Z=k]\,\mathbb{E}\big[B \,\big|\, Z=k\big],$$

and by Theorem 26.2.3, we have $\mathbb{E}\big[B \,\big|\, Z=k\big] \ge \left\lfloor \lg\binom{n}{k}\right\rfloor - 1$. Let $\varepsilon < p - 1/2$ be a constant to be determined shortly. For $n(p-\varepsilon) \le k \le n(p+\varepsilon)$, we have

$$\binom{n}{k} \ge \binom{n}{\lfloor n(p+\varepsilon)\rfloor} \ge \frac{2^{n\mathbb{H}(p+\varepsilon)}}{n+1},$$

by Corollary 26.1.5 (iii). We have

$$\begin{aligned}
\mathbb{E}[B] &\geq \sum_{k=\lfloor n(p-\varepsilon)\rfloor}^{\lceil n(p-\varepsilon)\rceil} \mathbb{P}[Z=k]\,\mathbb{E}\big[B \,\big|\, Z=k\big] \geq \sum_{k=\lfloor n(p-\varepsilon)\rfloor}^{\lceil n(p-\varepsilon)\rceil} \mathbb{P}[Z=k]\left(\left\lfloor \lg \binom{n}{k} \right\rfloor - 1\right) \\
&\geq \sum_{k=\lfloor n(p-\varepsilon)\rfloor}^{\lceil n(p-\varepsilon)\rceil} \mathbb{P}[Z=k]\left(\lg \frac{2^{n\mathbb{H}(p+\varepsilon)}}{n+1} - 2\right) \\
&= (n\mathbb{H}(p+\varepsilon) - \lg(n+1))\,\mathbb{P}[|Z-np| \leq \varepsilon n] \\
&\geq (n\mathbb{H}(p+\varepsilon) - \lg(n+1))\left(1 - 2\exp\left(-\frac{n\varepsilon^2}{4p}\right)\right),
\end{aligned}$$

since $\mu = \mathbb{E}[Z] = np$ and $\mathbb{P}\left[|Z-np| \geq \frac{\varepsilon}{p}pn\right] \leq 2\exp\left(-\frac{np}{4}\left(\frac{\varepsilon}{p}\right)^2\right) = 2\exp\left(-\frac{n\varepsilon^2}{4p}\right)$, by the Chernoff inequality. In particular, fix $\varepsilon > 0$, such that $\mathbb{H}(p+\varepsilon) > (1-\delta/4)\mathbb{H}(p)$, and since p is fixed $n\mathbb{H}(p) = \Omega(n)$, in particular, for n sufficiently large, we have $-\lg(n+1) \geq -\frac{\delta}{10}n\mathbb{H}(p)$. Also, for n sufficiently large, we have $2\exp\left(-\frac{n\varepsilon^2}{4p}\right) \leq \frac{\delta}{10}$. Putting it together, we have that for n large enough, we have

$$\mathbb{E}[B] \geq \left(1 - \frac{\delta}{4} - \frac{\delta}{10}\right)n\mathbb{H}(p)\left(1 - \frac{\delta}{10}\right) \geq (1-\delta)n\mathbb{H}(p),$$

as claimed.

As for the upper bound, observe that if an input sequence x has probability q, then the output sequence $y = \mathtt{Ext}(x)$ has probability to be generated which is at least q. Now, all sequences of length $|y|$ have equal probability to be generated. Thus, we have the following (trivial) inequality $2^{|\mathtt{Ext}(x)|}q \leq 2^{|\mathtt{Ext}(x)|}\,\mathbb{P}[y = \mathtt{Ext}(X)] \leq 1$, implying that $|\mathtt{Ext}(x)| \leq \lg(1/q)$. Thus,

$$\mathbb{E}[B] = \sum_x \mathbb{P}[X=x]\,|\mathtt{Ext}(x)| \leq \sum_x \mathbb{P}[X=x] \lg \frac{1}{\mathbb{P}[X=x]} = \mathbb{H}(X). \qquad \blacksquare$$

26.3. Bibliographical Notes

The presentation here follows [MU05, Sec. 9.1-Sec 9.3].

Chapter 27

Entropy II

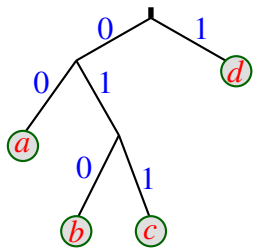
598 - Class notes for Randomized Algorithms
Sariel Har-Peled
December 10, 2019

> The memory of my father is wrapped up in white paper, like sandwiches taken for a day at work. Just as a magician takes towers and rabbits out of his hat, he drew love from his small body, and the rivers of his hands overflowed with good deeds.
>
> Yehuda Amichai, My Father

27.1. Huffman coding

A ***binary code*** assigns a string of 0s and 1s to each character in the alphabet. A code assigns for each symbol in the input a codeword over some other alphabet. Such a coding is necessary, for example, for transmitting messages over a wire, were you can send only 0 or 1 on the wire (i.e., for example, consider the good old telegraph and Morse code). The receiver gets a binary stream of bits and needs to decode the message sent. A prefix code, is a code where one can decipher the message, a character by character, by reading a prefix of the input binary string, matching it to a code word (i.e., string), and continuing to decipher the rest of the stream. Such a code is a ***prefix code***.

A binary code (or a prefix code) is ***prefix-free*** if no code is a prefix of any other. ASCII and Unicode's UTF-8 are both prefix-free binary codes. Morse code is a binary code (and also a prefix code), but it is not prefix-free; for example, the code for `S` ($\cdots$) includes the code for `E` ($\cdot$) as a prefix. (Hopefully the receiver knows that when it gets $\cdots$ that it is extremely unlikely that this should be interpreted as `EEE`, but rather `S`.



Any prefix-free binary code can be visualized as a binary tree with the encoded characters stored at the leaves. The code word for any symbol is given by the path from the root to the corresponding leaf; 0 for left, 1 for right. The length of a codeword for a symbol is the depth of the corresponding leaf. Such trees are usually referred to as ***prefix trees*** or ***code trees***.

The beauty of prefix trees (and thus of prefix odes) is that decoding is easy. As a concrete example, consider the tree on the right. Given a string '010100', we can traverse down the tree from the root, going left if get a '0' and right if we get '1'. Whenever we get to a leaf, we output the character output in the leaf, and we jump back to the root for the next character we are about to read. For the example '010100', after reading '010' our traversal in the tree leads us to the leaf marked with 'b', we jump back to the root and read the next input digit, which is '1', and this leads us to the leaf marked with 'd', which we output, and jump back to the root. Finally,

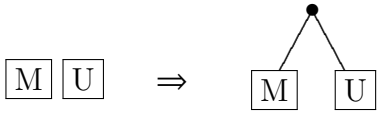
'00' leads us to the leaf marked by 'a', which the algorithm output. Thus, the binary string '010100' encodes the string `"bda"`.

Suppose we want to encode messages in an n-character alphabet so that the encoded message is as short as possible. Specifically, given an array frequency counts $f[1 \ldots n]$, we want to compute a prefix-free binary code that minimizes the total encoded length of the message. That is we would like to compute a tree T that minimizes

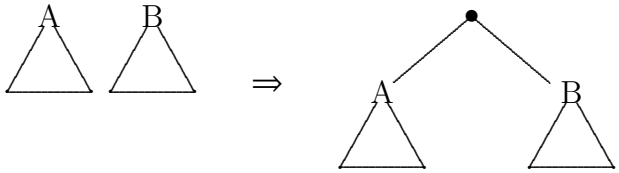
$$\text{cost}(T) = \sum_{i=1}^{n} f[i] * \text{len}(\text{code}(i)), \tag{27.1}$$

where $\text{code}(i)$ is the binary string encoding the ith character and $\text{len}(s)$ is the length (in bits) of the binary string s.

A nice property of this problem is that given two trees for some parts of the alphabet, we can easily put them together into a larger tree by just creating a new node and hanging the trees from this common node. For example, putting two characters together, we have the following.



Similarly, we can put together two subtrees.



27.1.1. The algorithm to build Hoffman's code

This suggests a simple algorithm that takes the two least frequent characters in the current frequency table, merge them into a tree, and put the merged tree back into the table (instead of the two old trees). The algorithm stops when there is a single tree. The intuition is that infrequent characters would participate in a large number of merges, and as such would be low in the tree – they would be assigned a long code word.

This algorithm is due to David Huffman, who developed it in 1952. Shockingly, this code is the best one can do. Namely, the resulting code is *asymptotically* gives the best possible compression of the data (of course, one can do better compression in practice using additional properties of the data and careful hacking). This ***Huffman coding*** is used widely and is the basic building block used by numerous other compression algorithms.

27.1.2. Analysis

Lemma 27.1.1. *Let T be an optimal code tree. Then T is a full binary tree (i.e., every node of T has either 0 or 2 children). In particular, if the height of T is d, then there are leafs nodes of height d that are sibling.*

Proof: If there is an internal node in T that has one child, we can remove this node from T, by connecting its only child directly with its parent. The resulting code tree is clearly a better compressor, in the sense of Eq. (27.1).

As for the second claim, consider a leaf u with maximum depth d in T, and consider it parent $v = \overline{\mathrm{p}}(u)$. The node v has two children, and they are both leafs (otherwise u would not be the deepest node in the tree), as claimed. ∎

Lemma 27.1.2. *Let x and y be the two least frequent characters (breaking ties between equally frequent characters arbitrarily). There is an optimal code tree in which x and y are siblings.*

Proof: More precisely, there is an optimal code in which x and y are siblings and have the largest depth of any leaf. Indeed, let T be an optimal code tree with depth d. The tree T has at least two leaves at depth d that are siblings, by Lemma 27.1.1.

Now, suppose those two leaves are not x and y, but some other characters α and β. Let $\mathcal{U}$ be the code tree obtained by swapping x and α. The depth of x increases by some amount Δ, and the depth of α decreases by the same amount. Thus,

$$\mathrm{cost}(\mathcal{U}) = \mathrm{cost}(T) - (f[\alpha] - f[x])\Delta.$$

By assumption, x is one of the two least frequent characters, but α is not, which implies that $f[\alpha] > f[x]$. Thus, swapping x and α does not increase the total cost of the code. Since T was an optimal code tree, swapping x and α does not decrease the cost, either. Thus, $\mathcal{U}$ is also an optimal code tree (and incidentally, $f[\alpha]$ actually equals $f[x]$). Similarly, swapping y and b must give yet another optimal code tree. In this final optimal code tree, x and y as maximum-depth siblings, as required. ∎

Theorem 27.1.3. *Huffman codes are optimal prefix-free binary codes.*

Proof: If the message has only one or two different characters, the theorem is trivial. Otherwise, let $f[1 \ldots n]$ be the original input frequencies, where without loss of generality, $f[1]$ and $f[2]$ are the two smallest. To keep things simple, let $f[n+1] = f[1] + f[2]$. By the previous lemma, we know that some optimal code for $f[1..n]$ has characters 1 and 2 as siblings. Let $\mathcal{T}_{\mathrm{opt}}$ be this optimal tree, and consider the tree formed by it by removing 1 and 2 as it leaves. We remain with a tree $\mathcal{T}'_{\mathrm{opt}}$ that has as leafs the characters $3, \ldots, n$ and a "special" character $n+1$ (which is the parent of 1 and 2 in $\mathcal{T}_{\mathrm{opt}}$) that has frequency $f[n+1]$. Now, since $f[n+1] = f[1] + f[2]$, we have

$$\begin{aligned}
\mathrm{cost}(\mathcal{T}_{\mathrm{opt}}) &= \sum_{i=1}^{n} f[i]\mathrm{depth}_{\mathcal{T}_{\mathrm{opt}}}(i) \\
&= \sum_{i=3}^{n+1} f[i]\mathrm{depth}_{\mathcal{T}_{\mathrm{opt}}}(i) + f[1]\mathrm{depth}_{\mathcal{T}_{\mathrm{opt}}}(1) + f[2]\mathrm{depth}_{\mathcal{T}_{\mathrm{opt}}}(2) - f[n+1]\mathrm{depth}_{\mathcal{T}_{\mathrm{opt}}}(n+1) \\
&= \mathrm{cost}\left(\mathcal{T}'_{\mathrm{opt}}\right) + (f[1] + f[2])\mathrm{depth}(\mathcal{T}_{\mathrm{opt}}) - (f[1] + f[2])(\mathrm{depth}(\mathcal{T}_{\mathrm{opt}}) - 1) \\
&= \mathrm{cost}\left(\mathcal{T}'_{\mathrm{opt}}\right) + f[1] + f[2]. \qquad (27.2)
\end{aligned}$$

This implies that minimizing the cost of $\mathcal{T}_{\mathrm{opt}}$ is equivalent to minimizing the cost of $\mathcal{T}'_{\mathrm{opt}}$. In particular, $\mathcal{T}'_{\mathrm{opt}}$ must be an optimal coding tree for $f[3 \ldots n+1]$. Now, consider the Huffman tree $\mathcal{U}_H$ constructed for $f[3, \ldots, n+1]$ and the overall Huffman tree T_H constructed for $f[1, \ldots, n]$. By the way the construction algorithm works, we have that $\mathcal{U}_H$ is formed by removing the leafs of 1 and 2 from T. Now, by induction, we know that the Huffman tree generated for $f[3, \ldots, n+1]$ is optimal; namely, $\mathrm{cost}\left(\mathcal{T}'_{\mathrm{opt}}\right) = \mathrm{cost}(\mathcal{U}_H)$. As such, arguing as above, we have

$$\mathrm{cost}(T_H) = \mathrm{cost}(\mathcal{U}_H) + f[1] + f[2] = \mathrm{cost}\left(\mathcal{T}'_{\mathrm{opt}}\right) + f[1] + f[2] = \mathrm{cost}(\mathcal{T}_{\mathrm{opt}}),$$

by Eq. (27.2). Namely, the Huffman tree has the same cost as the optimal tree. ∎

27.1.3. A formula for the average size of a code word

Assume that our input is made out of n characters, where the ith character is p_i fraction of the input (one can think about p_i as the probability of seeing the ith character, if we were to pick a random character from the input).

Now, we can use these probabilities instead of frequencies to build a Huffman tree. The natural question is what is the length of the codewords assigned to characters as a function of their probabilities?

In general this question does not have a trivial answer, but there is a simple elegant answer, if all the probabilities are power of 2.

Lemma 27.1.4. *Let $1, \ldots, n$ be n symbols, such that the probability for the ith symbol is p_i, and furthermore, there is an integer $l_i \geq 0$, such that $p_i = 1/2^{l_i}$. Then, in the Huffman coding for this input, the code for i is of length l_i.*

Proof: The proof is by easy induction of the Huffman algorithm. Indeed, for $n = 2$ the claim trivially holds since there are only two characters with probability $1/2$. Otherwise, let i and j be the two characters with lowest probability. It must hold that $p_i = p_j$ (otherwise, $\sum_k p_k$ can not be equal to one). As such, Huffman's merges this two letters, into a single "character" that have probability $2p_i$, which would have encoding of length $l_i - 1$, by induction (on the remaining $n - 1$ symbols). Now, the resulting tree encodes i and j by code words of length $(l_i - 1) + 1 = l_i$, as claimed. ∎

In particular, we have that $l_i = \lg 1/p_i$. This implies that the average length of a code word is

$$\sum_i p_i \lg \frac{1}{p_i}.$$

If we consider X to be a random variable that takes a value i with probability p_i, then this formula is

$$\mathbb{H}(X) = \sum_i \mathbb{P}[X = i] \lg \frac{1}{\mathbb{P}[X = i]},$$

which is the ***entropy*** of X.

Theorem 27.1.5. *Consider an input sequence S of m characters, where the characters are taken from an alphabet set Σ of size n. In particular, let f_i be the number of times the ith character of Σ appears in S, for $i = 1, \ldots, n$. Consider the compression of this string using Huffman's code. Then, the total length of the compressed string (ignoring the space needed to store the code itself) is $\leq m\,(\mathbb{H}(X) + 1)$, where X is a random variable that returns i with probability $p_i = f_i/m$.*

Proof: The trick is to replace p_i, which might not be a power of 2, by $q_i = 2^{\lfloor \lg p_i \rfloor}$. We have that $q_i \leq p_i \leq 2q_i$, and q_i is a power of 2, for all i. The leftover of this coding is $\Delta = 1 - \sum_i q_i$. We write Δ as a sum of powers of 2 (since the frequencies are fractions of the form i/m [since the input string is of length m] – this requires at most $\tau = O(\log m)$ numbers): $\Delta = \sum_{j=n+1}^{n+\tau} q_j$. We now create a Huffman code T for the frequencies $q_1, \ldots, q_n, q_{n+1}, \ldots, q_{n+\tau}$. The output length to encode the input string using this code, by Lemma 27.1.4, is

$$L = m \sum_{i=1}^{n} p_i \lg \frac{1}{q_i} \leq m \sum_{i=1}^{n} p_i \left(1 + \lg \frac{1}{p_i}\right) \leq m + m \sum_{i=1}^{n} p_i \lg \frac{1}{p_i} = m + m\mathbb{H}(X).$$

One can now restrict T to be a prefix tree only for the first n symbols. Indeed, delete the τ "fake" leafs/symbols, and repeatedly remove internal nodes that have only a single child. In the end of this process, we get a valid prefix tree for the first n symbols, and encoding the input string using this tree would require at most L bits, since process only shortened the code words. Finally, let $\mathcal{V}$ be the resulting tree.

Now, consider the Huffman tree code for the n input symbols using the original frequencies $p_1, \ldots p_n$. The resulting tree $\mathcal{U}$ is a better encoder for the input string than $\mathcal{V}$, by Theorem 27.1.3. As such, the compressed string, would have at most L bits – thus establishing the claim. ■

27.2. Compression

In this section, we consider the problem of how to compress a binary string. We map each binary string, into a new string (which is hopefully shorter). In general, by using a simple counting argument, one can show that no such mapping can achieve real compression (when the inputs are adversarial). However, the hope is that there is an underling distribution on the inputs, such that some strings are considerably more common than others.

Definition 27.2.1. A compression function `Compress` takes as input a sequence of n coin flips, given as an element of $\{H, T\}^n$, and outputs a sequence of bits such that each input sequence of n flips yields a distinct output sequence.

The following is easy to verify.

Lemma 27.2.2. *If a sequence S_1 is more likely than S_2 then the compression function that minimizes the expected number of bits in the output assigns a bit sequence to S_2 which is at least as long as S_1.*

Note, that this is weak. Usually, we would like the function to output a prefix code, like the Huffman code.

Theorem 27.2.3. *Consider a coin that comes up heads with probability $p > 1/2$. For any constant $\delta > 0$, when n is sufficiently large, the following holds.*

(i) There exists a compression function `Compress` *such that the expected number of bits output by* `Compress` *on an input sequence of n independent coin flips (each flip gets heads with probability p) is at most $(1+\delta)n\mathbb{H}(p)$; and*

(ii) The expected number of bits output by any compression function on an input sequence of n independent coin flips is at least $(1-\delta)n\mathbb{H}(p)$.

Proof: Let $\varepsilon > 0$ be a constant such that $p - \varepsilon > 1/2$. The first bit output by the compression procedure is '1' if the output string is just a copy of the input (using $n+1$ bits overall in the output), and '0' if it is compressed. We compress only if the number of ones in the input sequence, denoted by X is larger than $(p-\varepsilon)n$. By the Chernoff inequality, we know that $\mathbb{P}[X < (p-\varepsilon)n] \leq \exp\left(-n\varepsilon^2/2p\right)$.

If there are more than $(p-\varepsilon)n$ ones in the input, and since $p - \varepsilon > 1/2$, we have that

$$\sum_{j=\lceil n(p-\varepsilon)\rceil}^{n} \binom{n}{j} \leq \sum_{j=\lceil n(p-\varepsilon)\rceil}^{n} \binom{n}{\lceil n(p-\varepsilon)\rceil} \leq \frac{n}{2} 2^{n\mathbb{H}(p-\varepsilon)},$$

by Corollary 26.1.5. As such, we can assign each such input sequence a number in the range $0 \ldots \frac{n}{2} 2^{n\mathbb{H}(p-\varepsilon)}$, and this requires (with the flag bit) $1 + \lfloor \lg n + n\mathbb{H}(p-\varepsilon) \rfloor$ random bits.

Thus, the expected number of bits output is bounded by

$$(n+1)\exp\left(-n\varepsilon^2/2p\right) + (1 + \lfloor \lg n + n\mathbb{H}(p-\varepsilon)\rfloor) \leq (1+\delta)n\mathbb{H}(p),$$

by carefully setting ε and n being sufficiently large. Establishing the upper bound.

As for the lower bound, observe that at least one of the sequences having exactly $\tau = \lfloor (p+\varepsilon)n \rfloor$ heads, must be compressed into a sequence having

$$\lg \binom{n}{\lfloor (p+\varepsilon)n \rfloor} - 1 \geq \lg \frac{2^{n\mathbb{H}(p+\varepsilon)}}{n+1} - 1 = n\mathbb{H}(p-\varepsilon) - \lg(n+1) - 1 = \mu,$$

by Corollary 26.1.5. Now, any input string with less than τ heads has lower probability to be generated. Indeed, for a specific strings with $\alpha < \tau$ ones the probability to generate them is $p^\alpha(1-p)^{n-\alpha}$ and $p^\tau(1-p)^{n-\tau}$, respectively. Now, observe that

$$p^\alpha(1-p)^{n-\alpha} = p^\tau(1-p)^{n-\tau}\cdot\frac{(1-p)^{\tau-\alpha}}{p^{\tau-\alpha}} = p^\tau(1-p)^{n-\tau}\left(\frac{1-p}{p}\right)^{\tau-\alpha} < p^\tau(1-p)^{n-\tau},$$

as $1-p < 1/2 < p$ implies that $(1-p)/p < 1$.

As such, Lemma 27.2.2 implies that all the input strings with less than τ ones, must be compressed into strings of length at least μ, by an optimal compresser. Now, the Chenroff inequality implies that $\mathbb{P}[X \leq \tau] \geq 1 - \exp\left(-n\varepsilon^2/12p\right)$. Implying that an optimal compresser outputs on average at least $\left(1-\exp\left(-n\varepsilon^2/12p\right)\right)\mu$. Again, by carefully choosing ε and n sufficiently large, we have that the average output length of an optimal compressor is at least $(1-\delta)n\mathbb{H}(p)$. ∎

27.3. Bibliographical Notes

The presentation here follows [MU05, Sec. 9.1-Sec 9.3].

Chapter 28

Approximate Max Cut

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
December 10, 2019

We had encountered in the previous lecture examples of using rounding techniques for approximating discrete optimization problems. So far, we had seen such techniques when the relaxed optimization problem is a linear program. Interestingly, it is currently known how to solve optimization problems that are considerably more general than linear programs. Specifically, one can solve ***convex programming***. Here the feasible region is convex. How to solve such an optimization problems is outside the scope of this course. It is however natural to ask what can be done if one assumes that one can solve such general continuous optimization problems exactly.

In the following, we show that (optimization problem) max cut can be relaxed into a weird continuous optimization problem. Furthermore, this semi-definite program can be solved exactly efficiently. Maybe more surprisingly, we can round this continuous solution and get an improved approximation.

28.1. Problem Statement

Given an undirected graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ and nonnegative weights ω_{ij}, for all $ij \in \mathsf{E}$, the ***maximum cut problem*** (MAX CUT) is that of finding the set of vertices S that maximizes the weight of the edges in the cut $(S, \overline{S})$; that is, the weight of the edges with one endpoint in S and the other in $\overline{S}$. For simplicity, we usually set $\omega_{ij} = O$ for $ij \notin \mathsf{E}$ and denote the weight of a cut $\left(S, \overline{S}\right)$ by $w\left(S, \overline{S}\right) = \sum_{i \in S, j \in \overline{S}} \omega_{ij}$.

This problem is NP-Complete, and hard to approximate within a certain constant.

Given a graph with vertex set $\mathsf{V} = \{1, \ldots, n\}$ and nonnegative weights ω_{ij}, the weight of the maximum cut $w(S, \overline{S})$ is given by the following integer quadratic program:

$$\begin{aligned} \text{(Q)} \qquad & \max \quad \frac{1}{2} \sum_{i<j} \omega_{ij}(1 - y_i y_j) \\ & \text{subject to:} \quad y_i \in \{-1, 1\} \qquad \forall i \in \mathsf{V}. \end{aligned}$$

Indeed, set $S = \left\{ i \;\middle|\; y_i = 1 \right\}$. Clearly, $\omega\left(S, \overline{S}\right) = \frac{1}{2} \sum_{i<j} \omega_{ij}(1 - y_i y_j)$.

Solving quadratic integer programming is of course NP-Hard. Thus, we will relax it, by thinking about the numbers y_i as unit vectors in higher dimensional space. If so, the multiplication of the two vectors, is now replaced by dot product. We have:

$$\text{(P)} \qquad \max \quad \gamma = \frac{1}{2} \sum_{i<j} \omega_{ij} \left(1 - \left\langle v_i, v_j \right\rangle\right)$$

$$\text{subject to:} \quad v_i \in \mathbb{S}^{(n)} \qquad \forall i \in V,$$

where $\mathbb{S}^{(n)}$ is the n dimensional unit sphere in $\mathbb{R}^{n+1}$. This is an instance of semi-definite programming, which is a special case of convex programming, which can be solved in polynomial time (solved here means approximated within a factor of $(1+\varepsilon)$ of optimal, for any arbitrarily small $\varepsilon > 0$, in polynomial time). Namely, the solver finds a feasible solution with a the target function being arbitrarily close to the optimal solution. Observe that (P) is a relaxation of (Q), and as such the optimal solution of (P) has value larger than the optimal value of (Q).

The intuition is that vectors that correspond to vertices that should be on one side of the cut, and vertices on the other sides, would have vectors which are faraway from each other in (P). Thus, we compute the optimal solution for (P), and we uniformly generate a random vector $\mathbf{r}$ on the unit sphere $\mathbb{S}^{(n)}$. This induces a hyperplane h which passes through the origin and is orthogonal to $\mathbf{r}$. We next assign all the vectors that are on one side of h to S, and the rest to $\overline{S}$.

Summarizing, the algorithm is as follows: First, we solve (P), next, we pick a random vector $\mathbf{r}$ uniformly on the unit sphere $\mathbb{S}^{(n)}$. Finally, we set

$$S = \{v_i \mid \langle v_i, \mathbf{r}\rangle \geq 0\}.$$

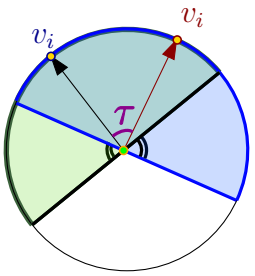
28.1.1. Analysis

The intuition of the above rounding procedure, is that with good probability, vectors in the solution of (P) that have large angle between them would be separated by this cut.

Lemma 28.1.1. *We have* $\mathbb{P}\big[\operatorname{sign}(\langle v_i, \mathbf{r}\rangle) \neq \operatorname{sign}(\langle v_j, \mathbf{r}\rangle)\big] = \frac{1}{\pi}\arccos(\langle v_i, v_j\rangle)$.

Proof: Let us think about the vectors v_i, v_j and $\mathbf{r}$ as being in the plane.

To see why this is a reasonable assumption, consider the plane g spanned by v_i and v_j, and observe that for the random events we consider, only the direction of $\mathbf{r}$ matter, which can be decided by projecting $\mathbf{r}$ on g, and normalizing it to have length 1. Now, the sphere is symmetric, and as such, sampling $\mathbf{r}$ randomly from $\mathbb{S}^{(n)}$, projecting it down to g, and then normalizing it, is equivalent to just choosing uniformly a vector from the unit circle.



Now, $\operatorname{sign}(\langle v_i, \mathbf{r}\rangle) \neq \operatorname{sign}(\langle v_j, \mathbf{r}\rangle)$ happens only if $\mathbf{r}$ falls in the double wedge formed by the lines perpendicular to v_i and v_j. The angle of this double wedge is exactly the angle between v_i and v_j. Now, since v_i and v_j are unit vectors, we have $\langle v_i, v_j\rangle = \cos(\tau)$, where $\tau = \angle v_i v_j$.

Thus,

$$\mathbb{P}\Big[\operatorname{sign}(\langle v_i, \mathbf{r}\rangle) \neq \operatorname{sign}(\langle v_j, \mathbf{r}\rangle)\Big] = \frac{2\tau}{2\pi} = \frac{1}{\pi}\arccos(\langle v_i, v_j\rangle),$$

as claimed. ∎

Theorem 28.1.2. *Let W be the random variable which is the weight of the cut generated by the algorithm. We have*

$$\mathbb{E}[W] = \frac{1}{\pi}\sum_{i<j}\omega_{ij}\arccos(\langle v_i, v_j\rangle).$$

Proof: Let X_{ij} be an indicator variable which is 1 if and only if the edge ij is in the cut. We have

$$\mathbb{E}\big[X_{ij}\big] = \mathbb{P}\Big[\operatorname{sign}(\langle v_i, \mathbf{r}\rangle) \neq \operatorname{sign}(\langle v_j, \mathbf{r}\rangle)\Big] = \frac{1}{\pi}\arccos(\langle v_i, v_j\rangle),$$

by Lemma 28.1.1. Clearly, $W = \sum_{i<j} \omega_{ij} X_{ij}$, and by linearity of expectation, we have

$$\mathbb{E}[W] = \sum_{i<j} \omega_{ij}\, \mathbb{E}\big[X_{ij}\big] = \frac{1}{\pi}\sum_{i<j} \omega_{ij} \arccos(\langle v_i, v_j\rangle). \qquad \blacksquare$$

Lemma 28.1.3. *For $-1 \le y \le 1$, we have $\dfrac{\arccos(y)}{\pi} \ge \alpha \cdot \dfrac{1}{2}(1-y)$, where*

$$\alpha = \min_{0\le\psi\le\pi} \frac{2}{\pi}\frac{\psi}{1-\cos(\psi)}. \tag{28.1}$$

Proof: Set $y = \cos(\psi)$. The inequality now becomes $\frac{\psi}{\pi} \ge \alpha\frac{1}{2}(1-\cos\psi)$. Reorganizing, the inequality becomes $\frac{2}{\pi}\frac{\psi}{1-\cos\psi} \ge \alpha$, which trivially holds by the definition of α. $\blacksquare$

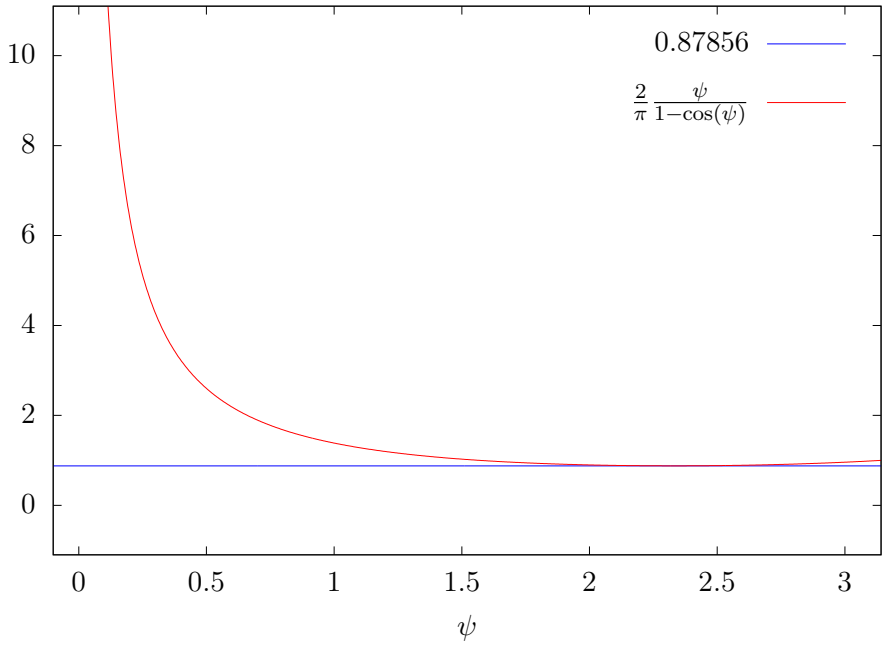


Figure 28.1: The function of Eq. (28.1).

Lemma 28.1.4. $\alpha > 0.87856$.

Proof: Using simple calculus, one can see that α achieves its value for $\psi = 2.331122...$, the nonzero root of $\cos\psi + \psi\sin\psi = 1$. $\blacksquare$

Theorem 28.1.5. *The above algorithm computes in expectation a cut with total weight $\alpha \cdot \mathrm{Opt} \ge 0.87856\mathrm{Opt}$, where* Opt *is the weight of the maximum weight cut.*

Proof: Consider the optimal solution to (P), and lets its value be $\gamma \ge \mathrm{Opt}$. We have

$$\mathbb{E}[W] = \frac{1}{\pi}\sum_{i<j} \omega_{ij} \arccos(\langle v_i, v_j\rangle) \ge \sum_{i<j} \omega_{ij}\alpha\frac{1}{2}(1-\langle v_i, v_j\rangle) = \alpha\gamma \ge \alpha\cdot\mathrm{Opt},$$

by Lemma 28.1.3. $\blacksquare$

28.2. Semi-definite programming

Let us define a variable $x_{ij} = \langle v_i, v_j \rangle$, and consider the n by n matrix M formed by those variables, where $x_{ii} = 1$ for $i = 1, \ldots, n$. Let V be the matrix having $v_1, \ldots, v_n$ as its columns. Clearly, $M = V^T V$. In particular, this implies that for any non-zero vector $v \in \mathbb{R}^n$, we have $v^T M v = v^T A^T A v = (Av)^T (Av) \geq 0$. A matrix that has this property, is called ***positive semidefinite***. Interestingly, any positive semidefinite matrix P can be represented as a product of a matrix with its transpose; namely, $P = B^T B$. Furthermore, given such a matrix P of size $n \times n$, we can compute B such that $P = B^T B$ in $O(n^3)$ time. This is know as ***Cholesky decomposition***.

Observe, that if a semidefinite matrix $P = B^T B$ has a diagonal where all the entries are one, then B has columns which are unit vectors. Thus, if we solve (P) and get back a semi-definite matrix, then we can recover the vectors realizing the solution, and use them for the rounding.

In particular, (P) can now be restated as

$$\begin{aligned} (SD) \qquad \max \quad & \frac{1}{2} \sum_{i<j} \omega_{ij} (1 - x_{ij}) \\ \text{subject to:} \quad & x_{ii} = 1 && \text{for } i = 1, \ldots, n \\ & (x_{ij})_{i=1,\ldots,n, j=1,\ldots,n} \text{ is a positive semi-definite matrix.} \end{aligned}$$

We are trying to find the optimal value of a linear function over a set which is the intersection of linear constraints and the set of positive semi-definite matrices.

Lemma 28.2.1. *Let $\mathcal{U}$ be the set of $n \times n$ positive semidefinite matrices. The set $\mathcal{U}$ is convex.*

Proof: Consider $A, B \in \mathcal{U}$, and observe that for any $t \in [0, 1]$, and vector $v \in \mathbb{R}^n$, we have:

$$v^T (tA + (1-t)B)v = v^T (tAv + (1-t)Bv) = tv^T Av + (1-t)v^T Bv \geq 0 + 0 \geq 0,$$

since A and B are positive semidefinite. ∎

Positive semidefinite matrices corresponds to ellipsoids. Indeed, consider the set $x^T Ax = 1$: the set of vectors that solve this equation is an ellipsoid. Also, the eigenvalues of a positive semidefinite matrix are all non-negative real numbers. Thus, given a matrix, we can in polynomial time decide if it is positive semidefinite or not (by computing the eigenvalues of the matrix).

Thus, we are trying to optimize a linear function over a convex domain. There is by now machinery to approximately solve those problems to within any additive error in polynomial time. This is done by using the interior point method, or the ellipsoid method. See [BV04, GLS93] for more details. The key ingredient that is required to make these methods work, is the ability to decide in polynomial time, given a solution, whether its feasible or not. As demonstrated above, this can be done in polynomial time.

28.3. Bibliographical Notes

The approximation algorithm presented is from the work of Goemans and Williamson [GW95]. Håstad [Hås01b] showed that MAX CUT can not be approximated within a factor of $16/17 \approx 0.941176$. Recently, Khot *et al.* [KKMO04] showed a hardness result that matches the constant of Goemans and Williamson (i.e., one can not approximate it better than α, unless P = NP). However, this relies on two conjectures,

the first one is the "Unique Games Conjecture", and the other one is "Majority is Stablest". The "Majority is Stablest" conjecture was recently proved by Mossel *et al.* [MOO05]. However, it is not clear if the "Unique Games Conjecture" is true, see the discussion in [KKMO04].

The work of Goemans and Williamson was quite influential and spurred wide research on using SDP for approximation algorithms. For an extension of the MAX CUT problem where negative weights are allowed and relevant references, see the work by Alon and Naor [AN04].

Bibliography

[ABKU99] Y. Azar, A. Broder, A. Karlin, and E. Upfal. Balanced allocations. *SIAM J. Comput.*, 29(1):180–200, 1999.

[ABN08a] Ittai Abraham, Yair Bartal, and Ofer Neiman. Nearly tight low stretch spanning trees. In *Proc. 49th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), pages 781–790. IEEE Computer Society, 2008.

[ABN08b] Ittai Abraham, Yair Bartal, and Ofer Neiman. Nearly tight low stretch spanning trees. *CoRR*, abs/0808.2017, 2008.

[Aga04] P. K. Agarwal. Range searching. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 36, pages 809–838. CRC Press LLC, Boca Raton, FL, USA, 2nd edition, 2004.

[AI06] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Proc. 47th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), pages 459–468. IEEE Computer Society, 2006.

[AI08] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, 2008.

[AKPW95] N. Alon, R. M. Karp, D. Peleg, and D. West. A graph-theoretic game and its application to the k-server problem. *SIAM J. Comput.*, 24(1):78–100, February 1995.

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

[AN04] N. Alon and A. Naor. Approximating the cut-norm via grothendieck's inequality. In *Proc. 36th Annu. ACM Sympos. Theory Comput.* (STOC), pages 72–80, 2004.

[Aro98] S. Arora. Polynomial time approximation schemes for Euclidean TSP and other geometric problems. *J. Assoc. Comput. Mach.*, 45(5):753–782, September 1998.

[Bar96] Y. Bartal. Probabilistic approximations of metric space and its algorithmic application. In *Proc. 37th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), pages 183–193, 10 1996.

[Bar98] Y. Bartal. On approximating arbitrary metrics by tree metrics. In *Proc. 30th Annu. ACM Sympos. Theory Comput.* (STOC), pages 161–168, 1998.

[BK90] Andrei Z. Broder and Anna R. Karlin. Multilevel adaptive hashing. In David S. Johnson, editor, *Proc. 1th ACM-SIAM Sympos. Discrete Algs.* (SODA), pages 43–53. SIAM, 1990.

[BV04] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge, 2004.

[Cha01] B. Chazelle. *The Discrepancy Method: Randomness and Complexity*. Cambridge University Press, New York, 2001.

[CLRS01] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press / McGraw-Hill, 2001.

[CS00] S. Cho and S. Sahni. A new weight balanced binary search tree. *Int. J. Found. Comput. Sci.*, 11(3):485–513, 2000.

[DIIM04] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proc. 20th Annu. Sympos. Comput. Geom.* (SoCG), pages 253–262, 2004.

[DP09] Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.

[EEST08] Michael Elkin, Yuval Emek, Daniel A. Spielman, and Shang-Hua Teng. Lower-stretch spanning trees. *SIAM J. Comput.*, 38(2):608–628, 2008.

[EHS14] D. Eppstein, S. Har-Peled, and A. Sidiropoulos. On the greedy permutation and counting distances. manuscript, 2014.

[FRT04] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Sys. Sci.*, 69(3):485–497, 2004.

[GLS93] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin Heidelberg, 2nd edition, 1993.

[Gre69] W.R. Greg. *Why are Women Redundant?* Trübner, 1869.

[GRSS95] M. Golin, R. Raman, C. Schwarz, and M. Smid. Simple randomized algorithms for closest pair problems. *Nordic J. Comput.*, 2:3–27, 1995.

[Gup00] A. Gupta. *Embeddings of Finite Metrics*. PhD thesis, University of California, Berkeley, 2000.

[GW95] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. Assoc. Comput. Mach.*, 42(6):1115–1145, nov 1995.

[Har11] S. Har-Peled. *Geometric Approximation Algorithms*, volume 173 of *Math. Surveys & Monographs*. Amer. Math. Soc., Boston, MA, USA, 2011.

[Hås01a] J. Håstad. Some optimal inapproximability results. *J. Assoc. Comput. Mach.*, 48(4):798–859, jul 2001.

[Hås01b] J. Håstad. Some optimal inapproximability results. *J. Assoc. Comput. Mach.*, 48(4):798–859, 2001.

[HR15] Sariel Har-Peled and Benjamin Raichel. Net and prune: A linear time algorithm for Euclidean distance problems. *J. Assoc. Comput. Mach.*, 62(6):44:1–44:35, 12 2015.

[HW87] D. Haussler and E. Welzl. ε-nets and simplex range queries. *Discrete Comput. Geom.*, 2:127–151, 1987.

[IM98] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. 30th Annu. ACM Sympos. Theory Comput.* (STOC), pages 604–613, 1998.

[Ind01] P. Indyk. Algorithmic applications of low-distortion geometric embeddings. In *Proc. 42nd Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), pages 10–31, 2001. Tutorial.

[JL84] W. B. Johnson and J. Lindenstrauss. Extensions of lipschitz mapping into hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[Kel56] J. L. Kelly. A new interpretation of information rate. *Bell Sys. Tech. J.*, 35(4):917–926, jul 1956.

[KKMO04] S. Khot, G. Kindler, E. Mossel, and R. O'Donnell. Optimal inapproximability results for max cut and other 2-variable csps. In *Proc. 45th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), pages 146–154, 2004. To appear in SICOMP.

[KKT91] C. Kaklamanis, D. Krizanc, and T. Tsantilas. Tight bounds for oblivious routing in the hypercube. *Math. sys. theory*, 24(1):223–232, 1991.

[KLMN05] R. Krauthgamer, J. R. Lee, M. Mendel, and A. Naor. Measured descent: A new embedding method for finite metric spaces. *Geom. funct. anal. (GAFA)*, 15(4):839–858, 2005.

[KOR00] E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM J. Comput.*, 2(30):457–474, 2000.

[LM00] B. Laurent and P. Massart. Adaptive estimation of a quadratic functional by model selection. *Ann. Statist.*, 28(5):1302–1338, 10 2000.

[Mat99] J. Matoušek. *Geometric Discrepancy*, volume 18 of *Algorithms and Combinatorics*. Springer, 1999.

[Mat02] J. Matoušek. *Lectures on Discrete Geometry*, volume 212 of *Grad. Text in Math.* Springer, 2002.

[McD89] C. McDiarmid. *Surveys in Combinatorics*, chapter On the method of bounded differences. Cambridge University Press, 1989.

[MNP06] R. Motwani, A. Naor, and R. Panigrahi. Lower bounds on locality sensitive hashing. In *Proc. 22nd Annu. Sympos. Comput. Geom.* (SoCG), pages 154–157, 2006.

[MOO05] E. Mossel, R. O'Donnell, and K. Oleszkiewicz. Noise stability of functions with low influences invariance and optimality. In *Proc. 46th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), pages 21–30, 2005.

[MP80] J. I. Munro and M. Paterson. Selection and sorting with limited storage. *Theo. Comp. Sci.*, 12:315–323, 1980.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, Cambridge, UK, 1995.

[MS09] M. Mendel and C. Schwob. Fast c-k-r partitions of sparse graphs. *Chicago J. Theor. Comput. Sci.*, 2009, 2009.

[MU05] M. Mitzenmacher and U. Upfal. *Probability and Computing – randomized algorithms and probabilistic analysis*. Cambridge, 2005.

[Rab76] M. O. Rabin. Probabilistic algorithms. In J. F. Traub, editor, *Algorithms and Complexity: New Directions and Recent Results*, pages 21–39. Academic Press, Orlando, FL, USA, 1976.

[SA96] R. Seidel and C. R. Aragon. Randomized search trees. *Algorithmica*, 16:464–497, 1996.

[Sch79] Arnold Schönhage. On the power of random access machines. In Hermann A. Maurer, editor, *Proc. 6th Int. Colloq. Automata Lang. Prog.* (ICALP), volume 71 of *Lect. Notes in Comp. Sci.*, pages 520–529. Springer, 1979.

[Sei93] R. Seidel. Backwards analysis of randomized geometric algorithms. In J. Pach, editor, *New Trends in Discrete and Computational Geometry*, volume 10 of *Algorithms and Combinatorics*, pages 37–68. Springer-Verlag, 1993.

[Smi00] M. Smid. Closest-point problems in computational geometry. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 877–935. Elsevier, Amsterdam, The Netherlands, 2000.

[Ste12] E. Steinlight. Why novels are redundant: Sensation fiction and the overpopulation of literature. *ELH*, 79(2):501–535, 2012.

[VÖ3] Berthold Vöcking. How asymmetry helps load balancing. *J. Assoc. Comput. Mach.*, 50(4):568–589, jul 2003.

[VC71] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl.*, 16:264–280, 1971.

[WG75] H. W. Watson and F. Galton. On the probability of the extinction of families. *J. Anthrop. Inst. Great Britain*, 4:138–144, 1875.

Index