A second course in randomized algorithms

Nick Harvey
University of British Columbia

December 3, 2025

Contents

Chapter 20

A Warmup: Set Cover

20.1 Definition and Background

Suppose we have a family of sets

$$S_1, \ldots, S_m \qquad \text{where} \qquad \bigcup_{j=1}^{m} S_j = [n].$$

The set $[n]$ is called the ***ground set***; recall that $[n]$ is standard notation for $\{1, \ldots, n\}$. Clearly each set in the family satisfies $S_j \subseteq [n]$.

We would like to choose as few of those sets as possible, while preserving the property that their union equals $[n]$. In mathematical notation, the Set Cover problem is

$$\min \left\{ \, |C| \; : \; C \subseteq [m] \text{ and } \bigcup_{j \in C} S_j = [n] \, \right\}.$$

Let OPT to refer to this minimum value.

An algorithm for this problem is called an α-approximation if it always outputs a cover C satisfying $|C| \leq \alpha \cdot \mathrm{OPT}$. Clearly $\alpha \geq 1$. Here α could be a constant or it could depend on other parameters, like n or m.

Some known facts about the Set Cover problem.

- It is NP-hard, as was shown by Karp in 1972. Thus, it is unlikely that any polynomial time algorithm can guarantee to produce an exact solution.
- The natural *greedy algorithm* is a $\ln n$-approximation algorithm.
- For every constant $c < 1$, it is very unlikely that there is a $(c \ln n)$-approximation algorithm. (If such an algorithm existed, this would have similar consequences to P=NP.) Thus, the greedy algorithm is the best possible.

Instead of discussing the greedy algorithm, we will discuss this problem with a viewpoint of mathematical optimization.

Integer program. The first step is to write the problem as an integer program (IP). For each $j \in [m]$, we create a Boolean variable x_j that indicates whether we select the set S_j. Using these variables, we can write the problem as

$$\begin{array}{lll} \min & \sum_{j=1}^{m} x_j & \\ \text{s.t.} & \sum_{j\,:\,i \in S_j} x_j \quad \geq 1 & \forall i \in [n] \\ & x_j \in \{0,1\} & \forall j \in [m] \end{array}$$

The objective function $\sum_{j=1}^{m} x_j$ counts the number of chosen sets. The i^{th} inequality constraint ensures that at least one of the chosen sets contains element i. Thus, this integer program captures the Set Cover problem exactly. It follows that its minimum value is also OPT, and it is also NP-hard to solve exactly.

Linear program. The next idea is to relax the integer program to a ***linear program*** (LP), by allowing the variables x_j to lie in the interval $[0,1]$ rather than the discrete set $\{0,1\}$. The resulting LP can be written

$$\begin{array}{lll} \min & \sum_{j=1}^{m} x_j & \\ \text{s.t.} & \sum_{j\,:\,i \in S_j} x_j \quad \geq 1 & \forall i \in [n] \\ & 0 \leq x_j \leq 1 & \forall j \in [m] \end{array}$$

We will use LPOPT to denote the minimum value of this LP.

Question 20.1.1. Does the value of the LP always equal the value of the IP? That is, does LPOPT always equal OPT?

Answer.

No. If it were true, then we'd have proven P=NP, because LPs can be solved in polynomial time. That is not an airtight argument, but we'll see an example below where they differ.

Question 20.1.2. Do we always have LPOPT $\leq$ OPT or LPOPT $\geq$ OPT?

Answer.

We always have LPOPT $\leq$ OPT. To see this, note that every feasible solution x to the IP is also a feasible solution to the LP. Thus, the LP is minimizing over a larger set of feasible solutions, and therefore its value can only be smaller.

Example 20.1.3. Consider the following example with $n = m = 3$.

$$S_1 = \{1,2\} \qquad S_2 = \{2,3\} \qquad S_3 = \{1,3\}\,.$$

It is clear that we must choose at least two sets in order to cover all three elements of the ground set. However, if we set $x_1 = x_2 = x_3 = 0.5$, then one can verify that this is a feasible solution to the LP with objective value 1.5.

Problem to Ponder 20.1.4. In this example, what is LPOPT, the optimal value of the LP?

The term ***integrality gap*** is used to refer to the ratio of the optimum values of an IP and its corresponding LP. Example 20.1.3 shows that our LP for Set Cover has an integrality gap greater than 1. Nevertheless, the LP is still useful. We will show two results.

1. The integrality gap of this LP is at most $\approx \ln n$.
2. Given any feasible solution x to the LP, there is a randomized algorithm that, with constant probability, can produce a valid set cover C whose size is at most $\approx \ln(n) \cdot \sum_j x_j$.

Question 20.1.5. Do you see why the second assertion implies the first?

Answer.

Consider an optimum solution to the LP. It has objective value at most LPOPT. Applying the randomized algorithm, we obtain a set cover satisfying $|C| \lessapprox \ln(n)\text{LPOPT}$. Since the best Set Cover solution is no larger than C, it follows that $\text{OPT} \lessapprox \ln(n)\text{LPOPT}$, which gives the desired bound on the integrality gap.

20.2 Randomized Rounding

In this section we design an algorithm that will prove the second assertion above. This is called a ***rounding algorithm***, because it converts a fractional solution into an integral solution.

Algorithm 20.1 Rounding the Set Cover LP. Assume that the input x is a feasible solution to the LP.

```
1: function SetCoverRound($x$)
2:     Let $C \leftarrow \emptyset$                    ▷ The indices of the chosen sets
3:     Let $L \leftarrow \ln(4n)$                      ▷ The rounding has $L$ phases
4:     for $t = 1, \ldots, L$
5:         for $j = 1, \ldots, m$
6:             Add $j$ to $C$ independently with probability $x_j$
7:     return $C$
8: end function
```

Lemma 20.2.1. The output C fails to cover all elements in $[n]$ (i.e., $\bigcup_{j \in C} S_j \neq [n]$) with probability at most $1/4$.

Claim 20.2.2. Consider the t^{th} phase of the algorithm. Each element $i \in [n]$ fails to be covered by the sets chosen in this phase with probability at most $1/e$.

Proof. Element i is not covered if every set containing i fails to be chosen. The probability of this is

$$\begin{aligned}
\Pr[\,(j \text{ not chosen})\ \forall j \text{ s.t. } i \in S_j\,] &= \prod_{j\,:\,i \in S_j} \Pr[\,j \text{ not chosen}\,] \qquad \text{(by independence)} \\
&= \prod_{j\,:\,i \in S_j} (1 - x_j) \\
&< \prod_{j\,:\,i \in S_j} \exp(-x_j) \qquad \text{(by Fact A.2.5)} \\
&= \exp\Big(-\sum_{j\,:\,i \in S_j} x_j\Big).
\end{aligned}$$

This is at most $1/e$ since x is feasible for the LP. $\square$

Proof of Lemma 20.2.1. First let us analyze the probability that a particular element is not covered by C. For any $i \in [n]$,

$$\begin{aligned}
\Pr[\text{element } i \text{ not covered}] &= \prod_{t=1}^{L} \Pr[\text{element } i \text{ not covered in phase } t] \\
&\leq \prod_{t=1}^{L} (1/e) \qquad \text{(by Claim 20.2.2)} \\
&= \exp(-L) \;=\; \frac{1}{4n} \qquad \text{(since } L = \ln(4n)\text{)}.
\end{aligned}$$

By a union bound (Fact A.3.8), the probability that *any* element fails to be covered by C is

$$\Pr[\,\textit{any} \text{ element not covered by } C\,] \;\leq\; \sum_{i=1}^{n} \Pr[\text{element } i \text{ not covered by } C] \;\leq\; \sum_{i=1}^{n} \frac{1}{4n} \;=\; \frac{1}{4}. \qquad \square$$

Lemma 20.2.3. The output C has $|C| > 4L \cdot \sum_j x_j$ with probability at most $1/4$.

Proof. In each phase, set j is added to C with probability x_j, so the expected number of sets added is $\sum_j x_j$. The expected number of edges added in *all* iterations is at most L times larger. That is, $\mathrm{E}[\,|C|\,] \leq L \cdot \sum_j x_j$. It follows that

$$\begin{aligned}
\Pr\left[\,|C| \geq 4L \cdot \textstyle\sum_j x_j\,\right] &\leq \frac{\mathrm{E}[\,|C|\,]}{4L \cdot \sum_j x_j} \qquad \text{(by Markov's inequality, Fact A.3.23)} \\
&\leq \frac{L \cdot \sum_j x_j}{4L \cdot \sum_j x_j} \;=\; \frac{1}{4}.
\end{aligned} \qquad \square$$

We conclude with the following theorem.

Theorem 20.2.4. The output C covers all elements and has size at most $4L \cdot \sum_j x_j$ with probability at least $1/2$.

References: (Shmoys and Shmoys, 2010, Section 1.7).

Proof. By Lemma 20.2.1, C fails to cover all elements with probability at most $1/4$. By Lemma 20.2.3, C has size exceeding $4L \cdot \sum_j x_j$ with probability at most $1/4$. By a union bound, the probability that either of these occurs is at most $1/2$. $\square$

Corollary 20.2.5. There is a randomized, polynomial-time algorithm that gives a $4\ln(4n)$-approximation to the Set Cover problem.

Proof. Compute an optimum solution x to the linear program; it satisfies $\sum_j x_j = \mathrm{LPOPT}$. This can be done in polynomial time, for example by interior point methods. By Theorem 20.2.4, the rounding algorithm above has probability $1/2$ of producing a Set Cover solution C with

$$|C| \;\leq\; 4L \cdot \sum_j x_j \;=\; 4L \cdot LPOPT \;\leq\; 4L \cdot OPT. \qquad \square$$

20.3 Exercises

Exercise 20.1. By adjusting the parameters, show that the integrality gap is at most $(1+2\epsilon)\ln(n/\epsilon)$ for any $\epsilon > 0$.

Exercise 20.2. Consider an instance of the Set Cover problem in which all sets have size $|S_i| \leq k$. Our randomized rounding algorithm gives an $O(\log n)$ approximation. However, it is known that the greedy algorithm gives a $\ln k$ approximation.

Can we modify the rounding algorithm to use fewer than $\ln n$ phases? Perhaps we can use only $\ln k$ phases and thereby get a $\ln k$ approximation? Unfortunately not.

Part I. Fix $k = 1$. Describe an instance of the Set Cover problem and a feasible LP solution such that, in each phase t, each element i has probability $p \geq 1/e - 1/4n$ of being uncovered in that phase.

Part II. Consider your instance from part (a). Let $\ell = \ln n - d$ for an arbitrary value d. Suppose we only perform the randomized rounding for ℓ phases. Prove that the expected number of elements who are still uncovered is $\Omega(e^d)$.

Part III. Consider the same instance from part (a). Let $\ell = \ln(n)/2$. Prove that, with probability at least $1 - 1/n$, there are at least $\sqrt{n}/2$ elements still uncovered after ℓ phases. You may assume that n is sufficiently large.

Chapter 21

Concentration Bounds, with details

21.1 Chernoff bound, in detail

The Chernoff bound was presented in a simplified form in Theorem 9.2.2. Let us now present it in a more elaborate form.

Let $X_1, \ldots, X_n$ be independent random variables such that X_i always lies in the interval $[0,1]$. Define $X = \sum_{i=1}^{n} X_i$. The expectation $\mathrm{E}[X]$ need not be exactly known, but we assume that $\check{\mu} \leq \mathrm{E}[X] \leq \hat{\mu}$. If it is exactly known, we may define $\check{\mu} = \hat{\mu} = \mathrm{E}[X]$.

Theorem 21.1.1. For all $\delta > 0$,

$$\begin{array}{llllll}
\text{Right tail:} & \Pr[\,X \geq (1+\delta)\hat{\mu}\,] \overset{(a)}{\leq} \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\hat{\mu}} \overset{(b)}{\leq} & \begin{cases} e^{-\delta^2\hat{\mu}/3} \\ e^{-(1+\delta)\ln(1+\delta)\hat{\mu}/4} \\ e^{-\delta\hat{\mu}/3} \end{cases} & \begin{array}{l} (\text{if } \delta \leq 1) \\ (\text{if } \delta \geq 1) \\ (\text{if } \delta \geq 1) \end{array} & \begin{array}{l} \text{“Gaussian tail”} \\ \text{“Poisson tail”} \\ \text{“Exponential tail”} \end{array} \\
\text{Left tail:} & \Pr[\,X \leq (1-\delta)\check{\mu}\,] \overset{(c)}{\leq} \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{\check{\mu}} \overset{(d)}{\leq} & e^{-\delta^2\check{\mu}/2} & & \text{“Gaussian tail”.}
\end{array}$$

Inequalities (c) and (d) are only valid for $\delta < 1$, but $\Pr[\,X \leq (1-\delta)\check{\mu}\,] = 0$ if $\delta > 1$.

References: (McDiarmid, 1998, Theorem 2.3), (Lehman et al., 2018, Theorem 20.5.1), (Motwani and Raghavan, 1995, Section 4.1), (Mitzenmacher and Upfal, 2005, equations (4.2) and (4.5)), (Klenke, 2008, Exercise 5.2.1), Wikipedia.

The tails have a qualitative difference in their dependence on δ.

- The “Gaussian tails” depend on δ^2. This resembles a Gaussian distribution (see Appendix B.4.3), whose tails look like $\exp(-\delta^2/2)$ (see Fact B.4.8).

- The “Poisson tail” depends on $\delta \ln \delta$. This resembles a Poisson distribution, whose mass function function looks like[1] $1/\delta! \approx \exp(-\delta \ln \delta)$.

- The “Exponential tail” depends on δ. This resembles the exponential distribution, whose tail looks like $e^{-\delta}$. This is weaker, but more convenient, than the “Poisson tail”.

[1]See, e.g., (Vershynin, 2018, Theorem 1.3.4) or these lecture notes.

21.2 Proofs for Chernoff Bound

21.2.1 Proof of inequality (a)

The Chernoff bounds would not be true without the assumption that $X_1, \ldots, X_n$ are independent. What special properties do independent random variables have? One basic property is that

$$\mathrm{E}[\, A \cdot B \,] \;=\; \mathrm{E}[\, A \,] \cdot \mathrm{E}[\, B \,] \tag{21.2.1}$$

for any independent random variables A and B.

References: (Lehman et al., 2018, Theorem 19.5.6), (Anderson et al., 2017, Fact 8.10), (Feller, 1968, Theorem IX.2.3), (Cormen et al., 2001, Exercise C.3-5), (Motwani and Raghavan, 1995, Proposition C.6), (Mitzenmacher and Upfal, 2005, Theorem 3.3), (Grimmett and Stirzaker, 2001, Lemma 3.3.9), (Durrett, 2019, Theorem 2.1.13), (Klenke, 2008, Theorem 5.4).

But the Chernoff bound has nothing to do with *products* of random variables, it is about *sums* of random variables. So one trick we could try is to convert sums into products using the exponential function. Fix some parameter $\theta > 0$ whose value we will choose later. We will look at the random variables

$$\begin{aligned} & \exp(\theta X_i) \qquad \forall i \in [n] \\ \text{and} \qquad & \exp(\theta X) \;=\; \exp\Big(\theta \sum_{i=1}^{n} X_i\Big) \;=\; \prod_{i=1}^{n} \exp(\theta X_i) \end{aligned}$$

Since $X_1, \ldots, X_n$ are independent, it follows[2] that $e^{\theta X_1}, \ldots, e^{\theta X_n}$ are also independent. Therefore, using (21.2.1) repeatedly,

$$\mathrm{E}\left[\, e^{\theta X} \,\right] \;=\; \prod_{i=1}^{n} \mathrm{E}\left[\, e^{\theta X_i} \,\right]. \tag{21.2.2}$$

So far this all seems promising. We want to prove that X is small, which is equivalent to proving that $e^{\theta X}$ is small. Using (21.2.2), we can do this by showing that each $\mathrm{E}\left[\, e^{\theta X_i} \,\right]$ is small. Perhaps we can somehow show $\mathrm{E}\left[\, e^{\theta X_i} \,\right]$ is small by comparing it to $\mathrm{E}[\, X_i \,]$?

If we were forgetting the rules of probability, we might be tempted to say that $\mathrm{E}\left[\, e^{\theta X_i} \,\right]$ equals $e^{\theta\, \mathrm{E}[\, X_i \,]}$, but that is false. We might remember one useful probability trick called Jensen's inequality (Fact B.4.2) that says $f(\mathrm{E}[\, A \,]) \leq \mathrm{E}[\, f(A) \,]$ for any random variable A and any convex function f. Applying this with $f(x) = e^{\theta x}$, we see that

$$e^{\theta\, \mathrm{E}[\, X_i \,]} \;\leq\; \mathrm{E}\left[\, e^{\theta X_i} \,\right]. \tag{21.2.3}$$

So we get a *lower bound* on $\mathrm{E}\left[\, e^{\theta X_i} \,\right]$ in terms of $\mathrm{E}[\, X_i \,]$, but we actually wanted an *upper bound*.

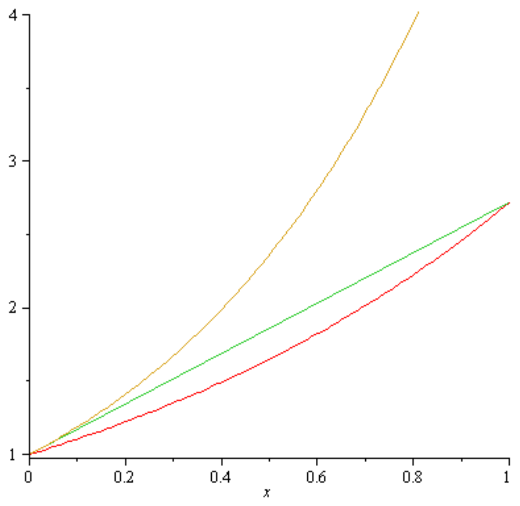
Claim 21.2.2 gives the desired upper bound; it shows that the inequality in (21.2.3) can almost be reversed. The proof is easy once we have the following convexity fact.

Claim 21.2.1. For all $\theta \in \mathbb{R}$ and all $x \in [0,1]$,

$$\exp(\theta x) \;\leq\; 1 + (e^{\theta} - 1)x \;\leq\; \exp\big((e^{\theta} - 1)x\big).$$

The inequalities are illustrated by this plot.

[2]See (Lehman et al., 2018, Lemma 19.2.2), (Cormen et al., 2001, Equation (C.24)), (Grimmett and Stirzaker, 2001, Theorem 3.2.3), (Klenke, 2008, Remark 2.15(iii)).



Proof of Claim 21.2.1. Consider the first inequality $e^{\theta x} \leq 1 + (e^\theta - 1)x$ for all $x \in [0,1]$. This follows from Fact B.3.8 by setting $c = e^\theta$. The second inequality $1 + (e^\theta - 1)x \leq \exp((e^\theta - 1)x)$ follows from the familiar Fact A.2.5. □

Claim 21.2.2. Let $\theta \in \mathbb{R}$ be arbitrary. Then $\mathrm{E}\left[\, e^{\theta X_i} \,\right] \leq \exp\left((e^\theta - 1)\,\mathrm{E}\left[\, X_i \,\right]\right)$.

Proof. The main idea is as follows. Although we cannot "pull the expectation inside the exponential", we can use Claim 21.2.1 to approximate the exponential by a linear function, then "pull the expectation inside" via linearity of expectation, then finally switch back to an exponential function.

The formal argument is

$$\mathrm{E}\left[\, e^{\theta X_i} \,\right] \;\leq\; \mathrm{E}\left[\, 1 + (e^\theta - 1)X_i \,\right] \;=\; 1 + (e^\theta - 1)\,\mathrm{E}\left[\, X_i \,\right] \;\leq\; \exp((e^\theta - 1)\,\mathrm{E}\left[\, X_i \,\right]),$$

where both inequalities follow from Claim 21.2.1. □

Now we are ready to prove the inequality (a) of the Chernoff bound.

$$\begin{aligned}
\Pr\left[\, X \geq (1+\delta)\hat{\mu} \,\right] &= \Pr\left[\, \exp(\theta X) \geq \exp\left(\theta(1+\delta)\hat{\mu}\right) \,\right] && \text{(by monotonicity)} \qquad (21.2.4)\\
&\leq \frac{\mathrm{E}\left[\, \exp(\theta X) \,\right]}{\exp(\theta(1+\delta)\hat{\mu})} && \text{(by Markov's inequality)}\\
&= \frac{\prod_{i=1}^n \mathrm{E}\left[\, e^{\theta X_i} \,\right]}{\exp(\theta(1+\delta)\hat{\mu})} && \text{(by (21.2.2))}\\
&\leq \frac{\prod_{i=1}^n \exp((e^\theta - 1)\,\mathrm{E}\left[\, X_i \,\right])}{\exp(\theta(1+\delta)\hat{\mu})} && \text{(by Claim 21.2.2).}
\end{aligned}$$

Gathering everything inside one exponential we get

$$\Pr\left[\, X \geq (1+\delta)\hat{\mu} \,\right] \;\leq\; \exp\left((e^\theta - 1)\sum_i \mathrm{E}\left[\, X_i \,\right] - \theta(1+\delta)\hat{\mu}\right). \qquad (21.2.5)$$

By calculus, the choice of θ that minimizes the right-hand side is

$$\theta \;=\; \ln(1+\delta). \qquad (21.2.6)$$

Plugging this into (21.2.5) and using $\sum_i \mathrm{E}[X_i] = \mathrm{E}[X] \leq \hat{\mu}$, we obtain

$$\Pr[X \geq (1+\delta)\hat{\mu}] \;\leq\; \exp\Big(\delta\hat{\mu} - \ln(1+\delta)(1+\delta)\hat{\mu}\Big),$$

which is equivalent to inequality (a).

References: Another exposition of inequality (a) can be found in (Lehman et al., 2018, Section 20.5.6).

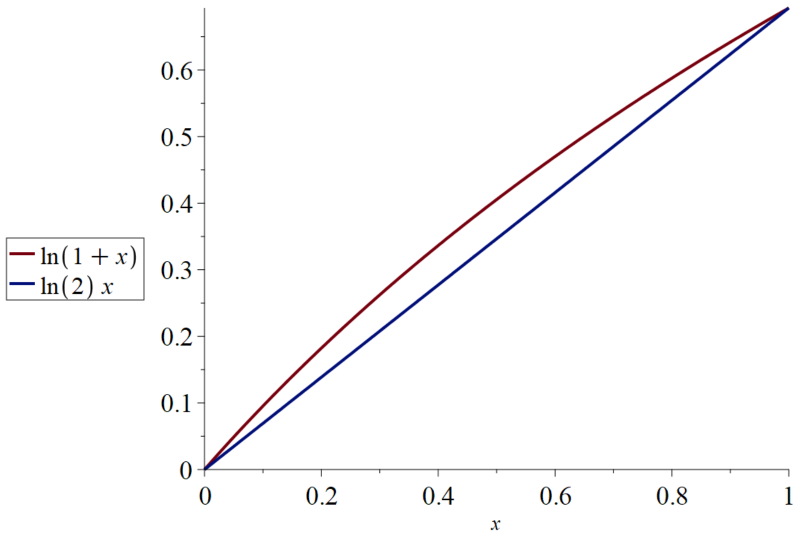
21.2.2 Proof of inequality (b), when $\delta \in [0,1]$

Claim 21.2.3. Then $(1+x)\ln(1+x) - x \geq \frac{\ln(2)}{2} \cdot x^2$ for all $x \in [0,1]$.

Proof. Note that the LHS and RHS both vanish at $x = 0$. So the claim holds if the derivative of the LHS is at least the derivative of the RHS on the interval $[0,1]$. By simple calculus,

$$\frac{d}{dx}\big[(1+x)\ln(1+x) - x\big] = \ln(1+x) \qquad \text{and} \qquad \frac{d}{dx}\frac{\ln(2)}{2}x^2 = \ln(2)x.$$

These derivatives are illustrated in the following figure.



They agree when $x = 0$ (both equal zero) and also agree when $x = 1$ (both equal $\ln 2$). Thus we have $\ln(1+x) \geq \ln(2)x$ for all $x \in [0,1]$, since the LHS is concave and the RHS is linear. □

Inequality (b) now follows straightforwardly because

$$\begin{aligned}
\frac{e^{\delta}}{(1+\delta)^{1+\delta}} &= \exp\Big(-\big((1+\delta)\ln(1+\delta) - \delta\big)\Big) \\
&\leq \exp\Big(-\frac{\ln 2}{2}\delta^2\Big) \qquad \text{(by Claim 21.2.3)} \\
&\leq e^{-\delta^2/3},
\end{aligned}$$

since $\ln(2) > 0.69 > 2/3$.

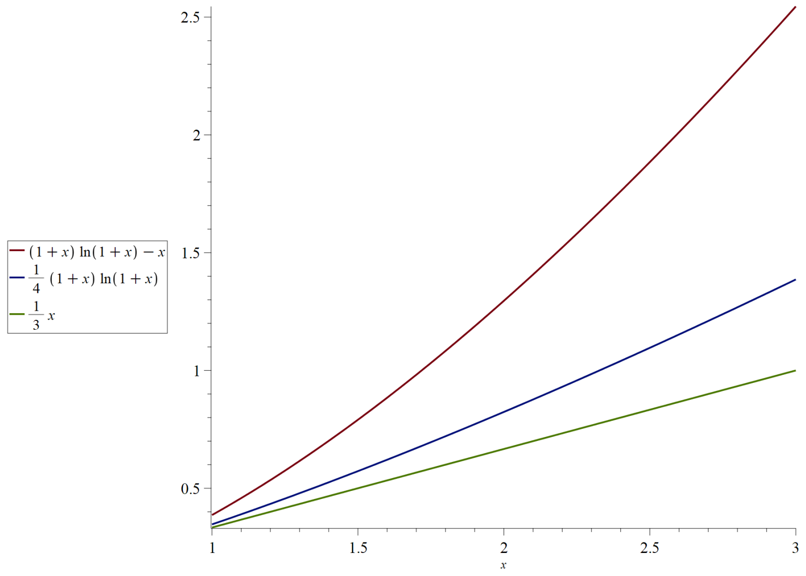
21.2.3 Proof of inequality (b), $\delta \geq 1$

Claim 21.2.4. Define

$$\begin{aligned} f(x) &= (1+x)\ln(1+x) - x \\ g(x) &= (1+x)\ln(1+x)/4 \\ h(x) &= x/3 \end{aligned}$$

Then $f(x) > g(x) > h(x)$ for $x \geq 1$.

This claim is illustrated by the following plot.



Proof. First we observe that the claimed inequality holds at the point $x = 1$. We have

$$\begin{aligned} f(1) &= 2\ln(2) - 1 \;>\; 0.38 \\ g(1) &= \ln(2)/4 \;\approx\; 0.346 \\ h(1) &= 1/3 \;<\; 0.34. \end{aligned}$$

We will show that their derivatives satisfy

$$\frac{d}{dx}f(x) \overset{(1)}{\geq} \frac{d}{dx}g(x) \overset{(2)}{\geq} \frac{d}{dx}h(x) \qquad \forall x \geq 1,$$

from which the claim follows by integration. These derivatives have the following expressions.

$$\frac{d}{dx}f(x) = \ln(1+x) \qquad \frac{d}{dx}g(x) = (1+\ln(1+x))/4 \qquad \text{and} \qquad \frac{d}{dx}h(x) = 1/3.$$

Inequality (2) is straightforward. For $x \geq 1$, we have $(1+\ln(1+x))/4 \geq (1+\ln(2))/4 \approx 0.423$, which is greater than $1/3$.

For inequality (1), first observe that $y \geq (1+y)/4$ for $y \geq 1/3$. Substituting $y \leftarrow \ln(1+x)$ gives $\ln(1+x) \geq (1+\ln(1+x))/4$ for $x \geq e^{1/3} - 1 \approx 0.395$. This implies (1). □

We can now show inequality (b) of Theorem 21.1.1 in the case $\delta \geq 1$. Using the notation of Claim 21.2.4,

$$\begin{aligned} \frac{e^\delta}{(1+\delta)^{1+\delta}} &= \exp\Big(-\big((1+\delta)\ln(1+\delta)-\delta\big)\Big) \\ &= \exp\big(-f(\delta)\big) \;<\; \exp\big(-g(\delta)\big) \;<\; \exp\big(-h(\delta)\big). \end{aligned}$$

Substitituting the definition of g and h, we obtain

$$\Big(\frac{e^\delta}{(1+\delta)^{1+\delta}}\Big)^{\hat{\mu}} \;<\; \exp\big(-(1+\delta)\ln(1+\delta)\hat{\mu}/4\big) \;<\; \exp\big(-\delta\hat{\mu}/3\big).$$

21.2.4 Proof of inequality (c)

The argument is very similar to the proof of inequality (a). We will choose θ to be negative.

$$\begin{aligned} \Pr[\,X \leq (1-\delta)\check{\mu}\,] &= \Pr\big[\exp(\theta X) \geq \exp\big(\theta(1-\delta)\check{\mu}\big)\,\big] && \text{(by monotonicity and } \theta<0) \\ &\leq \frac{\mathrm{E}[\exp(\theta X)\,]}{\exp(\theta(1-\delta)\check{\mu})} && \text{(by Markov's inequality)} \\ &\leq \frac{\prod_{i=1}^n \exp((e^\theta-1)\,\mathrm{E}[\,X_i\,])}{\exp(\theta(1-\delta)\check{\mu})} && \text{(by Claim 21.2.2)} \\ &= \exp\Big((e^\theta-1)\sum_{i=1}^n \mathrm{E}[\,X_i\,] - \theta(1-\delta)\check{\mu}\Big) \\ &\leq \exp\Big((e^\theta-1)\check{\mu} - \theta(1-\delta)\check{\mu}\Big). \end{aligned}$$

This last inequality holds because $\theta < 0$ so $e^\theta - 1 < 0$, and because $\check{\mu} \leq \sum_{i=1}^n \mathrm{E}[\,X_i\,]$. Plugging in $\theta = \ln(1-\delta)$, which is negative, we obtain

$$\Pr[\,X \leq (1-\delta)\check{\mu}\,] \;=\; \Big(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\Big)^{\check{\mu}},$$

which is inequality (c).

21.2.5 Proof of inequality (d)

The statement and the proof are similar to Claim 21.2.3.

Claim 21.2.5. Then $(1-x)\ln(1-x)+x \geq \frac{1}{2}\cdot x^2$ for all $x \in [0,1)$.

Proof. Note that the LHS and RHS both vanish at $x=0$. So the claim holds if the derivative of the LHS is at least the derivative of the RHS on the interval $[0,1)$. By simple calculus,

$$\frac{d}{dx}\big[(1-x)\ln(1-x)+x\big] = -\ln(1-x) \qquad \text{and} \qquad \frac{d}{dx}x^2/2 = x.$$

The linear approximation of $-\ln(1-x)$ at $x=0$ is

$$x \cdot \tfrac{d}{dx}\big(-\ln(1-x)\big)\Big|_{x=0} = x\cdot\big(\tfrac{1}{1-x}\big)\Big|_{x=0} = x.$$

Furthermore, $-\ln(1-x)$ is convex on $[0,1)$ because its second derivative is $1/(1-x)^2 \geq 0$. Thus $-\ln(1-x) \geq x$ on $[0,1)$. □

Inequality (d) now follows straightforwardly because, using Claim 21.2.5,

$$\frac{e^{-\delta}}{(1-\delta)^{1-\delta}} \;=\; \exp\Big(-\big((1-\delta)\ln(1-\delta)+\delta\big)\Big) \;\leq\; \exp\Big(-\delta^2/2\Big).$$

21.3 Proof of the Hoeffding Bound

The Hoeffding Bound was introduced in Section 9.3. We will prove the following slightly weaker result.

Theorem 21.3.1. Let $X_1, \ldots, X_n$ be independent random variables such that X_i always lies in the interval $[0,1]$. Define $X = \sum_{i=1}^{n} X_i$. Then

$$\Pr\left[\,|X - \mathrm{E}\left[\,X\,\right]| \geq t\,\right] \;\leq\; 2\exp(-t^2/2n) \qquad \forall t \geq 0.$$

Simplifications. First of all, we will "center" the random variables, which cleans up the inequality by eliminating the expectation. Define $\hat{X}_i = X_i - \mathrm{E}\left[\,X_i\,\right]$ and $\hat{X} = \sum_{i=1}^{n} \hat{X}_i$. Note that[3] $\hat{X}_i \in [-1,1]$. Our main argument is to prove that

$$\Pr\left[\hat{X} \geq t\right] \;\leq\; \exp(-t^2/2n). \tag{21.3.1}$$

The same argument also applies to $-\hat{X}$, so we get that

$$\Pr\left[-\hat{X} \geq t\right] \;=\; \Pr\left[\hat{X} \leq -t\right] \;\leq\; \exp(-t^2/2n).$$

Combining them with a union bound, we get

$$\Pr\left[\,|X - \mathrm{E}\left[\,X\,\right]| \geq t\,\right] \;=\; \Pr\left[|\hat{X}| \geq t\right] \;\leq\; \Pr\left[\hat{X} \geq t\right] + \Pr\left[-\hat{X} \geq t\right] \;\leq\; 2\exp(-t^2/2n).$$

This proves the theorem (with the weaker exponent).

Proof of (21.3.1). As in the proof of the Chernoff Bound (see (21.2.1)), we will use the fact that $\mathrm{E}\left[\,A \cdot B\,\right] = \mathrm{E}\left[\,A\,\right] \cdot \mathrm{E}\left[\,B\,\right]$ for any independent random variables A and B.

Key Idea #1: As in the proof of the Chernoff Bound, we will convert sums into products using the exponential function. Fix some parameter $\lambda > 0$ whose value we will choose later. Define

$$\begin{aligned} Y_i &= \exp(\lambda \hat{X}_i) \\ Y &= \exp(\lambda \hat{X}) \;=\; \exp\Big(\lambda \sum_{i=1}^{n} \hat{X}_i\Big) \;=\; \prod_{i=1}^{n} \exp(\lambda \hat{X}_i) \;=\; \prod_{i=1}^{n} Y_i. \end{aligned}$$

It is easy to check that, since $\{X_1, \ldots, X_n\}$ is mutually independent, so is $\left\{\hat{X}_1, \ldots, \hat{X}_n\right\}$ and $\{Y_1, \ldots, Y_n\}$. Therefore, as in (21.2.2),

$$\mathrm{E}\left[\,Y\,\right] \;=\; \prod_{i=1}^{n} \mathrm{E}\left[\,Y_i\,\right]. \tag{21.3.2}$$

[3]This step is where the argument is not careful enough to obtain the optimal exponent: $\hat{X}_i$ is actually supported on an interval of length 1, although our argument only assumes that it is supported on an interval of length 2.

So far this all seems quite good. We want to prove that $\hat{X}$ is small, which is equivalent to proving Y is small. Using (21.3.2), we can do this by showing that the $\mathrm{E}[\,Y_i\,]$ terms are small. Doing so involves an extremely useful tool.

Key Idea #2: The second main idea is a clever trick to bound terms of the form $\mathrm{E}[\,\exp(\lambda A)\,]$, where A is a mean-zero random variable. We discuss this idea in more detail in the next subsection. We will use[4] Lemma 21.3.2 to show

$$\mathrm{E}[\,Y_i\,] \;=\; \mathrm{E}\Big[\exp(\lambda\hat{X}_i)\Big] \;\le\; \exp(\lambda^2/2). \tag{21.3.3}$$

Thus, combining this with (21.3.2),

$$\mathrm{E}[\,Y\,] \;\le\; \prod_{i=1}^{n}\exp(\lambda^2/2) \;=\; \exp(\lambda^2 n/2). \tag{21.3.4}$$

Now we are ready to prove Hoeffding's inequality:

$$\begin{aligned}
\Pr\Big[\,\hat{X}\ge t\,\Big] \;&=\; \Pr\Big[\exp(\lambda\hat{X})\ge\exp\big(\lambda t\big)\Big] \qquad \text{(by monotonicity of } e^x\text{)}\\
&\le\; \frac{\mathrm{E}\Big[\exp(\lambda\hat{X})\Big]}{\exp(\lambda t)} \qquad \text{(by Markov's inequality)}\\
&=\; \mathrm{E}[\,Y\,]\cdot\exp(-\lambda t)\\
&\le\; \exp(\lambda^2 n/2-\lambda t) \qquad \text{(by (21.3.4))}\\
&=\; \exp(-t^2/2n),
\end{aligned}$$

by optimizing to get $\lambda = t/n$. □

21.3.1 Hoeffding's Lemma

The second main idea of Hoeffding's inequality is the following claim.

Lemma 21.3.2 (Hoeffding's Lemma, symmetric version). Let A be a random variable such that $|A|\le 1$ with probability 1 and $\mathrm{E}[\,A\,]=0$. Then for any $\lambda>0$, we have $\mathrm{E}[\,\exp(\lambda A)\,]\le\exp(\lambda^2/2)$.

Intuitively, the expectation should be maximized by the random variable A that is uniform on $\{-1,+1\}$. In this case,

$$\mathrm{E}[\,\exp(\lambda A)\,] \;=\; \frac{1}{2}e^{\lambda}-\frac{1}{2}e^{-\lambda} \;\le\; e^{\lambda^2/2}.$$

This inequality is a nice bound on the hyperbolic cosine function (Claim 21.3.3). The full proof of Lemma 21.3.2 basically reduces to the case of $A\in\{-1,1\}$ using convexity of e^x.

Proof. Define $p=(1+A)/2$ and $q=(1-A)/2$. Observe that $p,q\ge 0$, $p+q=1$, and $p-q=A$. By convexity,

$$\exp(\lambda A)=\exp\big(\lambda(p-q)\big)=\exp\big(\lambda p+(-\lambda)q\big)\le p\cdot\exp(\lambda)+q\cdot\exp(-\lambda)=\frac{e^{\lambda}+e^{-\lambda}}{2}+\frac{A}{2}(e^{\lambda}-e^{-\lambda}).$$

[4] If we were more careful here and instead used Lemma 21.3.4, we could improve the constant in the exponent in (21.3.3) from 1/2 to 1/8. This would improve the constant in the exponent in (21.3.1) from 1/2 to 2.

Thus,

$$\mathrm{E}\left[\exp(\lambda A)\right] \;\leq\; \mathrm{E}\left[\frac{e^{\lambda}+e^{-\lambda}}{2}+\frac{A}{2}(e^{\lambda}-e^{-\lambda})\right] \;=\; \frac{e^{\lambda}+e^{-\lambda}}{2},$$

since $\mathrm{E}[A]=0$. This last quantity is bounded by the following technical claim. ∎

Claim 21.3.3 (Approximation of Cosh). For any real x, we have $(e^x+e^{-x})/2 \leq \exp(x^2/2)$.

References: (Alon and Spencer, 2000, Lemma A.1.5).

Proof. First observe that the product of all the even numbers at most $2n$ does not exceed the product of all numbers at most $2n$. In symbols,

$$2^n(n!) \;=\; \prod_{i=1}^{n}(2i) \;\leq\; \prod_{i=1}^{2n} i \;=\; (2n)!$$

Now to bound $(e^x+e^{-x})/2$, we write it as a Taylor series and observe that the odd terms cancel.

$$\frac{e^x+e^{-x}}{2} \;=\; \sum_{n\geq 0}\frac{x^n}{n!}+\sum_{n\geq 0}\frac{(-x)^n}{n!} \;=\; \sum_{n\geq 0}\frac{x^{2n}}{(2n)!} \;\leq\; \sum_{n\geq 0}\frac{x^{2n}}{2^n(n!)} \;=\; \sum_{n\geq 0}\frac{(x^2/2)^n}{n!} \;=\; \exp(x^2/2)\; ∎$$

A common scenario is that A is mean-zero, but lies in an "asymmetric" interval $[a,b]$, where $a<0<b$. A tighter version of Lemma 21.3.2 can be derived for this scenario, although its proof is more complicated.

Lemma 21.3.4 (Hoeffding's Lemma, asymmetric version). Let A be a random variable such that $A \in [a,b]$ with probability 1 and $\mathrm{E}[A]=0$. Then for any $\lambda>0$, we have $\mathrm{E}[\exp(\lambda A)] \leq \exp\left(\lambda^2(b-a)^2/8\right)$.

References: (McDiarmid, 1998, Lemma 2.6), (Cesa-Bianchi and Lugosi, 2006, Lemma A.1), (Shalev-Shwartz and Ben-David, 2014, Lemma B.7), (Alon and Spencer, 2000, Theorem A.1.17), Wikipedia.

The proof uses ideas similar to the proof of Lemma 21.3.2, except we cannot use Claim 21.3.3 and must instead use an ad-hoc calculus argument.

21.3.2 The Generalized Hoeffding Inequality

Theorem 21.3.5 (Hoeffding's General Inequality). Let $X_1,\ldots,X_n$ be independent random variables where $X_i \in [a_i,b_i]$. Let $X=\sum_{i=1}^n X_i$. Then

$$\begin{aligned}
\text{Left tail:}\quad & \Pr\left[\sum_{i=1}^n X_i \leq \mathrm{E}[X]-s\right] && \leq\; \exp\left(-2\frac{s^2}{\sum_{i=1}^n(b_i-a_i)^2}\right)\\
\text{Right tail:}\quad & \Pr\left[\sum_{i=1}^n X_i \geq \mathrm{E}[X]+s\right] && \leq\; \exp\left(-2\frac{s^2}{\sum_{i=1}^n(b_i-a_i)^2}\right)\\
\text{Combined tails:}\quad & \Pr\left[\left|\sum_{i=1}^n X_i - \mathrm{E}[X]\right| \geq s\right] && \leq\; 2\exp\left(-2\frac{s^2}{\sum_{i=1}^n(b_i-a_i)^2}\right).
\end{aligned}$$

In particular, for any desired $q\in(0,1)$, setting $s=\sqrt{\ln(2/q)\sum_{i=1}^n(b_i-a_i)^2/2}$ gives

$$\Pr\left[\,|\textstyle\sum_i X_i - \mathrm{E}[X]| \geq s\,\right] \;\leq\; q.$$

References: (McDiarmid, 1998, Theorem 2.5), (Vershynin, 2018, Theorem 2.2.6), (Boucheron et al., 2012, Theorem 2.8), (Roch, 2020, Theorem 2.40), (Dubhashi and Panconesi, 2009, Problem 1.9), (Grimmett and Stirzaker, 2001, Theorem 12.2.3), Wikipedia.

Proof of Theorem 21.3.5. We will prove only the right tail. Let $\overline{X}_i = X_i - \mathrm{E}[\,X_i\,]$. For $\theta > 0$,

$$\begin{aligned}
\Pr\left[\,(\textstyle\sum_{i=1}^n X_i) - \mathrm{E}[\,X\,] \geq s\,\right] &= \Pr\left[\,\textstyle\sum_i \overline{X}_i \geq s\,\right] \\
&= \Pr\left[\exp(\theta \textstyle\sum_i \overline{X}_i) \geq e^{\theta s}\right] && \text{(monotonicity)} \\
&\leq \mathrm{E}\left[\,\exp(-\theta s + \theta \textstyle\sum_i \overline{X}_i)\,\right] && \text{(Markov inequality)} \\
&= e^{-\theta s} \prod_i \mathrm{E}\left[\,\exp(\theta \overline{X}_i)\,\right] && \text{(independence of } \overline{X}_i\text{)} \\
&\leq e^{-\theta s} \prod_i e^{\theta^2 (b_i - a_i)^2/8} && \text{(by Lemma 21.3.4)} \\
&= \exp\big(-\theta s + \theta^2 \textstyle\sum_i (b_i - a_i)^2/8\big) && (21.3.5)
\end{aligned}$$

By simple calculus, $\theta \to -\theta s + \theta^2 \alpha$ is minimized by taking $\theta = s/2\alpha$, and the minimum value is $-s^2/4\alpha$. Thus (21.3.5) is minimized by taking $\theta = 4s/\sum_i (b_i - a_i)^2$, yielding the bound

$$\Pr\left[\,\textstyle\sum_i X_i - \mathrm{E}[\,X_i\,] \geq s\,\right] \;\leq\; \exp\Big(-\frac{2s^2}{\sum_i (b_i - a_i)^2}\Big).$$

This is the claimed right tail. □

References: Dubhashi-Panconesi Problem 1.9, Ledoux-Talagrand Section 4.1, Wikipedia.

21.4 Exercises

Exercise 21.1 **Massart's Lemma.** Let $P \subset \mathbb{R}^d$ be a set of points satisfying $\|p\| \leq 1$ for all $p \in P$. (Here $\|p\|$ is the Euclidean norm $\sqrt{\sum_{i=1}^d p_i^2}$.) Let $m = |P|$. Let $\xi_1, \ldots, \xi_d \in \{-1, +1\}$ be uniform and independent random signs.

Part I. Use the Generalized Hoeffding Inequality to prove that, for all $s \geq 0$,

$$\Pr\left[\max_{p \in P} \sum_{i=1}^d p_i \xi_i \geq s\right] \;\leq\; m \cdot \exp(-s^2/2).$$

Part II. Using the previous part, prove that

$$\mathrm{E}\left[\max_{p \in P} \sum_{i=1}^d p_i \xi_i\right] \;\leq\; \sqrt{2 \ln m} + O(1).$$

Chapter 22

More Applications of Concentration

22.1 Balls and Bins: The Heaviest Bin

Let us return to the topic of balls and bins, which was first introduced in Chapter 7. Consider throwing n balls into n bins, uniformly and independently. Let B_i be the number of balls in bin i. In Section 7.6 we used an ad hoc argument to analyze the load on the heaviest bin, namely $\max_i B_i$. Now we will give an alternative analysis using the Chernoff bound.

Theorem 22.1.1. Assume $n \geq 3$. Define

$$\alpha \;=\; \frac{16 \ln n}{\ln \ln n}.$$

With probability at least $1 - 1/n$, the heaviest bin has at most α balls. That is,

$$\Pr\left[\max_i B_i \leq \alpha\right] \;\geq\; 1 - 1/n.$$

This theorem is optimal up to constant factors. It is known that $\max_i B_i \geq \ln n / \ln \ln n$ with probability at least $1 - 1/n$. See, e.g., (Mitzenmacher and Upfal, 2005, Lemma 5.12).

This choice of α is motivated by the following claim.

Claim 22.1.2. $\alpha \geq 2$ and $\alpha \ln \alpha \geq 8 \ln n$.

Proof. The first step is to give tail bounds on B_1. We decompose B_1 into indicator random variables as

$$B_1 \;=\; X_1 + X_2 + \cdots + X_n,$$

where X_j is 1 if the j^{th} ball lands in the first bin. For each $j \in [n]$ we have $\mathrm{E}[\,X_j\,] = 1/n$, so $\mathrm{E}[\,B_1\,] = \sum_j \mathrm{E}[\,X_j\,] = 1$. What is the probability that this first bin has at least α balls?

We will analyze this event using Theorem 21.1.1, inequality (b) for the case $\delta \geq 1$. Specifically, letting

$X = B_1$ and $\delta = \alpha - 1 \geq 1$, we obtain

$$\begin{aligned}
\Pr[\,B_1 \geq \alpha\,] &= \Pr[\,B_1 \geq \alpha \operatorname{E}[\,B_1\,]\,] && \text{(since } \operatorname{E}[\,B_1\,] = 1\text{)}\\
&\leq \exp\big(-\operatorname{E}[\,B_1\,] \cdot \alpha \ln(\alpha)/4\big) && \text{(by the Chernoff bound)}\\
&= \exp\big(-\alpha \ln(\alpha)/4\big) && \text{(since } \operatorname{E}[\,B_1\,] = 1\text{)}\\
&\leq \exp\big(-(8\ln n)/4\big) && \text{(by Claim 22.1.2)}\\
&= \frac{1}{n^2}. && (22.1.1)
\end{aligned}$$

So bin 1 is unlikely to have more than α balls.

Here we have analyzed bin 1, but the bins are all equivalent so the same analysis actually holds for all bins. The remainder of the argument is just the union bound.

$$\begin{aligned}
\Pr[\,\textit{any}\text{ bin has} \geq \alpha \text{ balls}\,] &= \Pr[\,B_1 \geq \alpha \,\vee\, B_2 \geq \alpha \,\vee\, \cdots \,\vee\, B_n \geq \alpha\,]\\
&\leq \sum_{i=1}^{n} \Pr[\,B_i \geq \alpha\,] && \text{(Fact A.3.8)}\\
&\leq \sum_{i=1}^{n} \frac{1}{n^2} && \text{(by (22.1.1))}\\
&= \frac{1}{n}.
\end{aligned}$$

Thus, with probability at least $1 - 1/n$, all bins have at most α balls. □

Proof of Claim 22.1.2. First we claim that $2\sqrt{x} > \ln x$ for all $x > 0$. For $x \in (0,1]$ this is clear because $2\sqrt{x} > 0 \geq \ln x$. In the regime $x > 1$ we have $1/\sqrt{x} > 1/x$, so integrating both sides shows that $2\sqrt{x} > \ln x$.

Applying this inequality with $x = \ln n$, we obtain $2\sqrt{\ln n} > \ln\ln n$. Since $n \geq 3$ we have $\ln\ln n > 0$, so the previous inequality may be rewritten as

$$2\frac{\ln n}{\ln\ln n} \;>\; \sqrt{\ln n}. \tag{22.1.2}$$

From this we may infer (again using $n \geq 3$) that

$$\alpha \;\geq\; 8 \cdot \Big(2\frac{\ln n}{\ln\ln n}\Big) \;\geq\; 8 \cdot \sqrt{\ln 3} \;>\; 2.$$

Next we take the log of both sides of (22.1.2), obtaining

$$\ln\alpha \;>\; \ln\Big(2\frac{\ln n}{\ln\ln n}\Big) \;\geq\; (1/2)\ln\ln n.$$

Thus, to conclude

$$\alpha\ln\alpha \;\geq\; 16\frac{\ln n}{\ln\ln n} \cdot \big((1/2)\ln\ln n\big) \;=\; 8\ln n,$$

which was the claimed bound. □

22.2 Congestion Minimization

One of the classically important areas in algorithm design and combinatorial optimization is *network flows*. A central problem in that area is the maximum flow problem. We now look at a generalization of this problem. An instance of the problem consists of a graph $G = (V, E)$ and a sequence $(s_1, t_1), \ldots, (s_k, t_k)$ of pairs of vertices. (The graph can be directed or undirected.) Let $n = V$ and $m = |E|$. For convenience, we will assume there are no multi-edges, and that commodities have distinct source-sink pairs, so that $m, k \leq n^2$.

A natural question to ask is: do there exist paths P_i from s_i to t_i for every i such that these paths share no arcs? This is called the **edge-disjoint paths** problem. For directed graphs, the problem is NP-hard even in the case $k = 2$. For undirected graphs, the problem is polynomial time solvable if k is a fixed constant, but NP-hard if k is a sufficiently large function of n.

We will look at a variant of this problem called the **congestion minimization** problem. The idea is to allow each edge to be used in multiple paths, but not too many paths. The number of paths using a given edge is called the "congestion" of that edge. We say that a solution has congestion C if it is a collection of paths P_i from s_i to t_i, where each edge is contained in at most C of the paths. The problem is to find the minimum value of C such that there is a solution of congestion C. This problem is still NP-hard, since determining if $C = 1$ is precisely the edge-disjoint paths problem.

We will look at the congestion minimization problem from the point of view of approximation algorithms. Let OPT be the minimum congestion of any solution. We would like to give an algorithm which can produce a solution with congestion at most $\alpha \cdot \mathrm{OPT}$ for some $\alpha \geq 1$. This factor α is the called the approximation factor of the algorithm.

Theorem 22.2.1. There is an algorithm for the congestion minimization problem with approximation factor $O(\log n / \log\log n)$.

To design such an algorithm we will use linear programming. We write down an integer program (IP) which captures the problem exactly, relax that to a linear program (LP), then design a method for "rounding" solutions of the LP into solutions for the IP.

The Integer Program Writing an IP formulation of an optimization problem is usually quite simple. That is indeed true for the congestion minimization problem. However, we will use an IP which you might find rather odd: our IP will have *exponentially many variables.* This will simplify our explanation of the rounding.

Let $\mathcal{P}^i$ be the set of *all* paths in G from s_i to t_i. (Note that $|\mathcal{P}^i|$ may be exponential in n.) For every path $P \in \mathcal{P}^i$, we create a variable x_P^i. This variable will take values only in $\{0, 1\}$, and setting it to 1 corresponds to including the path P in our solution.

The integer program is as follows.

$$\begin{array}{lllll}
\min & C & & & \\
\text{s.t.} & \displaystyle\sum_{P \in \mathcal{P}^i} x_P^i & = 1 & & \forall i \in [k] \\
& \displaystyle\sum_{i} \sum_{\substack{P \in \mathcal{P}^i \\ e \in P}} x_P^i & \leq C & & \forall e \in E \\
& C & \geq 1 & & \\
& x_P^i \in \{0, 1\} & & & \forall i \in [k],\ P \in \mathcal{P}^i
\end{array}$$

The last constraint says that we must decide for every path whether or not to include it in the solution.

The first constraint says that the solution must choose exactly one path between each pair s_i and t_i. The second constraint ensures that the number of paths using each arc is at most C. The optimization objective is to find the minimum value of C such that a solution exists. Lastly, we have the constraint $C \geq 1$. This seems redundant, since it is clearly satisfied by any solution: every path s_i-t_i must traverse some edge, forcing congestion at least 1. Nevertheless, it will be slightly helpful for the linear program below.

Every solution to the IP corresponds to a solution for the congestion minimization problem with congestion C, and vice-versa. Thus the optimum value of the IP is OPT — the minimum congestion of any solution to the original problem.

This integer program is NP-hard to solve, so we relax it into a linear program, by replacing the integrality constraints with non-negativity constraints. The resulting linear program is:

$$\begin{array}{lllr}
\min & C & & \\
\text{s.t.} & \displaystyle\sum_{P \in \mathcal{P}^i} x_P^i & = 1 & \forall i \in [k] \\
& \displaystyle\sum_i \sum_{\substack{P \in \mathcal{P}^i \\ e \in P}} x_P^i & \leq C & \forall e \in E \\
& C & \geq 1 & \\
& x_P^i & \geq 0 & \forall i \in [k],\ P \in \mathcal{P}^i
\end{array}$$

Using techniques[1] in combinatorial optimization, we can:

- solve this LP exactly in time $\text{poly}(n)$, and
- ensure that the number of non-zero variables x_P^i is $\text{poly}(n)$.

Let C^* be the optimal value of the LP. Recall that we have a constraint which ensures that $C^* \geq 1$.

Claim 22.2.2. $C^* \leq \text{OPT}$.

Proof. The LP was obtained from the IP by removing constraints, so the LP's feasible region contains the IP's feasible region. Thus, any feasible solution for the IP is also feasible for the LP. In particular, the optimal solution for the IP is feasible for the LP. So the LP has a solution with objective value equal to OPT, and its minimum objective value is at most OPT. $\square$

The Rounding. Our algorithm will solve the LP and most likely obtain a "fractional" solution — a solution with some non-integral variables, which is therefore not feasible for the IP. The next step of the algorithm is to "round" that fractional solution into a solution which is is feasible for the IP. In doing so, the congestion might increase, but we will ensure that it does not increase too much.

We will use *randomized rounding* in such a way that we choose *exactly one path* from s_i to t_i for each $i \in [k]$. That is, we generate independent random variables $P_1, \ldots, P_k$, where $P_i \in \mathcal{P}^i$, such that

$$\Pr[P_i = P] \; = \; x_P^i \qquad \forall i \in [k],\ P \in \mathcal{P}^i.$$

Observe that the LP's constraints ensure that these are indeed probabilities: for each i, $x_P^i \geq 0$ and $\sum_{P \in \mathcal{P}^i} x_P^i = 1$. The algorithm simply outputs the chosen paths $P_1, \ldots, P_k$.

[1] There are two ways to do this. The first way is to solve the dual LP using the ellipsoid method. This can be done in $\text{poly}(n)$ time even though it can have exponentially many constraints. The second way is to find a "compact formulation" of the LP which uses fewer variables, much like the usual LP that you may have seen for the ordinary maximum flow problem.

22.2.1 Analysis

Our main taks is to analyze the congestion of these chosen paths. Consider a specific edge $e \in E$. The number of chosen paths that use edge e is clearly

$$\sum_{i=1}^{k} \mathbf{1}_{e \in P_i}.$$

Furthermore, the random variables $\mathbf{1}_{e \in P_i}$ are independent because the chosen paths $P_1, \ldots, P_k$ are independent.

Claim 22.2.3. The expected number of chosen paths that use edge e is

$$\mathrm{E}\left[\sum_{i=1}^{k} \mathbf{1}_{e \in P_i}\right] \;\leq\; C^*.$$

Proof. Since the events "$P_i = P$" are disjoint, we have

$$\mathrm{E}\left[\,\mathbf{1}_{e \in P_i}\,\right] \;=\; \sum_{\substack{P \in \mathcal{P}^i \\ e \in P}} \Pr\left[\,P_i = P\,\right] \;=\; \sum_{\substack{P \in \mathcal{P}^i \\ e \in P}} x_P^i.$$

Thus, by linearity of expectation, the expected number of chosen paths using edge e is

$$\mathrm{E}\left[\sum_{i=1}^{k} \mathbf{1}_{e \in P_i}\right] \;=\; \sum_{i=1}^{k} \sum_{\substack{P \in \mathcal{P}^i \\ e \in P}} x_P^i.$$

The second constraint of the LP ensures that this is at most C. Furthermore, since we assume that our fractional solution is optimal for the LP, we have $C = C^*$. □

This claim only tells us about the *expected* congestion. When using randomized rounding, there will tend to be variance, and some edges might end up with a congestion exceeding this expectation. We will use the Chernoff bound to show that the congestion in unlikely to be too much larger. Specifically, define[2]

$$\alpha \;=\; \frac{24 \ln n}{\ln \ln n}.$$

We will show that it is very likely that *every* edge has congestion at most αC^*. This choice of α is motivated by the following claim, which is a trivial modification of Claim 22.1.2.

Claim 22.2.4. $\alpha \geq 2$ and $\alpha \ln \alpha \geq 12 \ln n$.

Lemma 22.2.5. Fix an edge e. Then

$$\Pr\left[\sum_{i=1}^{k} \mathbf{1}_{e \in P_i} \geq \alpha C^*\right] \;\leq\; \frac{1}{n^3}.$$

Proof. We will apply the Chernoff bound to

$$X = \sum_{i=1}^{k} \mathbf{1}_{e \in P_i}.$$

[2] We have $\alpha > 0$ due to the assumption that $n \geq 3$.

Note that X is a sum of independent random variables, each of which takes values in $\{0,1\}$. Define $\hat{\mu} = C^*$ and note that $\mathrm{E}[\,X\,] \leq \hat{\mu}$ by Claim 22.2.3. Next define $\delta = \alpha - 1$ and note that $\delta \geq 1$ by Claim 22.2.4.

Thus we may apply Theorem 21.1.1, inequality (b) for the case $\delta \geq 1$. This yields

$$\begin{aligned}
\Pr[\,X \geq \alpha C^*\,] &= \Pr[\,X \geq (1+\delta)\hat{\mu}\,] && \text{(definition of } \delta \text{ and } \hat{\mu}) \\
&\leq \exp\big(-(1+\delta)\ln(1+\delta)\hat{\mu}/4\big) && \text{(Chernoff bound)} \\
&= \exp\big(-(\alpha\ln\alpha)C^*/4\big) && \text{(definition of } \delta) \\
&\leq \exp(-3\ln n),
\end{aligned}$$

where this last inequality uses Claim 22.2.4 and also that $C^* \geq 1$ (due to the last constraint of the LP). □

Theorem 22.2.6. With probability at least $1 - 1/n$, the rounding produces a list of paths $P_1, \ldots, P_k$ with congestion at most αOPT.

Proof. The proof follows easily by a union bound. The probability that any edge has congestion at least C^* is exactly

$$\begin{aligned}
\Pr\left[\exists e \in E \text{ s.t. } \sum_{i=1}^{k} \mathbf{1}_{e\in P_i} \geq \alpha C^*\right] &\leq \sum_{e\in E} \Pr\left[\sum_{i=1}^{k} \mathbf{1}_{e\in P_i} \geq \alpha C^*\right] && \text{(by a union bound)} \\
&\leq \sum_{e\in E} (1/n^3) && \text{(by Lemma 22.2.5)} \\
&\leq 1/n,
\end{aligned}$$

since $|E| \leq n^2$. By Claim 22.2.2 we have $C^* \leq$ OPT. Thus, the probability that any edge has congestion at most αOPT is at most $1/n$. □

Further Remarks. The *rounding* algorithm that we presented is actually optimal: there are graphs for which $OPT/C^* = \Omega(\log n/\log\log n)$. Consequently, every rounding algorithm which converts a fractional solution of LP to an integral solution of IP must necessarily incur an increase of $\Omega(\log n/\log\log n)$ in the congestion.

That statement does not rule out the possibility that there is a better algorithm which behaves completely differently (i.e., one which does not use IP or LP at all). But sadly it turns out that there is no better algorithm (for the case of directed graphs). It is known that every efficient algorithm must have approximation factor $\alpha = \Omega(\log n/\log\log n)$, assuming a reasonable complexity theoretic conjecture (NP $\not\subseteq$ BPTIME$(n^{O(\log\log n)})$). So the algorithm that we presented is optimal, up to constant factors.

22.3 Error-correcting codes

Suppose a person (the sender) wants to transmit a message to another person (the receiver). This message might somehow be corrupted during transmission. The goal is for the receiver to determine the original message, even if these corruptions occur.

There are various ways to model corruptions, such as stochastic noise (or Shannon model), stochastic erasures, stochastic deletions, adversarial erasures, etc.

We will consider the ***adversarial noise model*** (or Hamming model), in which it is assumed that at most t symbols of the message are be corrupted. The values and locations of those corruptions can be arbitrary.

The sender and receiver will first agree upon a ***code***, which we define to be a set of binary strings of length n. These strings are called ***codewords***. In symbols, the ***Hamming cube*** is $\{0,1\}^n$, and a code is a set

$$\text{Code:} \qquad \mathcal{C} \subseteq \{0,1\}^n .$$

Since the codewords are binary strings, $\mathcal{C}$ is sometimes called a ***binary code***. The set $\mathcal{C}$ can be ordered, for example by viewing the codewords as binary numbers. Thus we may write $\mathcal{C} = \{C_1, \ldots, C_M\}$ for some value M. Note that $|\mathcal{C}| \leq 2^n$, so

$$2^n \;\geq\; M \tag{22.3.1}$$

The communication scheme is as follows. Suppose that there are M different messages that the sender could send. We can think of a message simply as being an integer $m \in [M]$. (For example, the messages might be arbitrary binary strings of length $\lg M$.) Each message m can be identified with a codeword in a canonical way; for example, message m corresponds to codeword C_m.

Transmitting m directly would take $\lceil \lg M \rceil$ bits. Instead, the sender transmits C_m, which takes n bits. This requires more bits to be transmitted:

$$n \;\geq\; \lceil \lg M \rceil \tag{22.3.2}$$

because of (22.3.1). This value n is called the ***block length*** of the code. The efficiency of the code is measured by how tight the inequality (22.3.2) is. Quantatively, the ***rate*** of $\mathcal{C}$ is defined to be

$$R \;=\; \frac{\lg |\mathcal{C}|}{n} \;=\; \frac{\lg M}{n}.$$

The main question: if the codeword is corrupted during transmission, can the receiver still determine the original message m?

22.3.1 Decoding

Let $\oplus$ denote Binary Xor. The ***Hamming distance*** between $x, y \in \{0,1\}^n$ is

$$\Delta(x,y) \;=\; \sum_{i=1}^{n} (x_i \oplus y_i) \;=\; \sum_{i=1}^{n} |x_i - y_i| \;=\; \|x-y\|_1 . \tag{22.3.3}$$

Observe that Δ satisfies the triangle inequality

$$\Delta(x,z) \;\leq\; \Delta(x,y) + \Delta(y,z) \tag{22.3.4}$$

because $\|\cdot\|_1$ is a norm. (This also follows by applying Fact A.2.4 component-wise.)

The ***minimum distance*** (or simply ***distance***) of $\mathcal{C}$ is

$$d \;=\; \min_{x \neq y \in \mathcal{C}} \Delta(x,y).$$

The ***relative distance*** is d/n.

Claim 22.3.1 (Unique decoding). Let $x \in \mathcal{C}$ be arbitrary. Produce y from x by flipping at most $\left\lfloor \frac{d-1}{2} \right\rfloor$ bits. Then, given y, one can determine x.

Proof. By construction of y, the Hamming distance $\Delta(x,y)$ satisfies

$$\Delta(x,y) \;\leq\; \left\lfloor \frac{d-1}{2} \right\rfloor.$$

However, for any other codeword $z \in \mathcal{C}$, where $z \neq x$, we have

$$\begin{aligned}
\Delta(z,y) &\geq \Delta(x,z) - \Delta(x,y) \qquad \text{(by the triangle inequality (22.3.4))}\\
&\geq d - \left\lfloor \frac{d-1}{2} \right\rfloor\\
&\geq d/2 + 1/2 \;>\; \left\lfloor \frac{d-1}{2} \right\rfloor.
\end{aligned}$$

We have shown that y's Hamming distance to x is smaller than to any other z. □

22.3.2 A random code construction

Theorem 22.3.2. For any $\epsilon > 0$, there exists a binary code of relative distance $1/2 - \epsilon$ and rate $\Omega(\epsilon^2)$.

References: MIT course notes.

Remark 22.3.3.

- You might wonder why we aim for relative distance nearly 1/2 instead of relative distance nearly 1. This is because binary codes with relative distance nearly 1 have rate going to zero with n. Specifically, if $\delta > 1/2 + \epsilon$ then $M \leq 1/\epsilon$ so the rate is $R \leq \lg(1/\epsilon)/n$, which vanishes as $n \to \infty$. See (Guruswami et al., 2019, Theorem 4.4.1).
- The Gilbert-Varshamov greedy code construction also achieves relative distance $1/2 - \epsilon$ and rate $\Omega(\epsilon^2)$. See (Guruswami et al., 2019, Theorem 4.2.1 and Proposition 3.3.5).
- For codes with relative distance $1/2 - \epsilon$, the rate of $\Omega(\epsilon^2)$ is nearly optimal. The so-called Linear Programming bound implies a $O(\epsilon^2 \log(1/\epsilon))$ upper bound on the rate. See (Guruswami et al., 2019, Section 8.2). For the special case of "balanced codes", the $O(\epsilon^2 \log(1/\epsilon))$ bound follows from an algebraic argument due to Alon.

Claim 22.3.4. Let y, z be independent and uniformly random points in $\{0,1\}^n$. Then

$$\Pr\left[\, \Delta(y,z)/n < 1/2 - \epsilon \,\right] \;\leq\; \exp(-\epsilon^2 n).$$

Proof. The Hamming distance $\Delta(y,z)$ can be decomposed into indicators as

$$\Delta(y,z) \;=\; \sum_{i=1}^{n} X_i$$

where $X_i = y_i \oplus z_i$. Note that X_i is an unbiased Bernoulli RV — it is 0 or 1 with probability 1/2. (This actually follows from Corollary A.3.26.) The expected distance is $\mu = \mathrm{E}\left[\, \Delta(y,z) \,\right] = n/2$. Then, by the Chernoff bound,

$$\begin{aligned}
\Pr\left[\, \Delta(y,z)/n < 1/2 - \epsilon \,\right] &= \Pr\left[\, \Delta(y,z) < (1-2\epsilon)\mu \,\right]\\
&< \exp\big(-(2\epsilon)^2 \mu/2 \big)\\
&= \exp(-\epsilon^2 n).
\end{aligned}$$

□

Proof of Theorem 22.3.2. Let $M = \exp(\epsilon^2 n/2)$ and pick $C_1, ..., C_M$ in $\{0,1\}^n$ independently and uniformly at random. The previous claim shows that, for any pair of these points, they are unlikely to have small Hamming distance. This is extended to all pairs of points by the union bound.

$$
\begin{aligned}
\Pr\left[\,\exists i,j \text{ s.t. } \Delta(C_i,C_j)/n < 1/2-\epsilon\,\right] &\leq \binom{M}{2}\cdot\exp(-\epsilon^2 n) && \text{(union bound)}\\
&< \frac{M^2}{2}\cdot\exp(-\epsilon^2 n) && \text{(by Claim 22.3.4)}\\
&= \frac{1}{2}\cdot\big(\underbrace{\exp(\epsilon^2 n/2)}_{=M}\big)^2\cdot\exp(-\epsilon^2 n)\\
&= 1/2.
\end{aligned}
$$

We define the code $\mathcal{C} = \{C_1, \ldots, C_M\}$. It definitely has rate

$$
\frac{\lg M}{n} \;=\; \frac{\epsilon^2 n/2}{\ln(2)n} \;=\; \Omega(\epsilon^2).
$$

Furthermore, with probability at least $1/2$, we have $\Delta(C_i,C_j)/n \geq 1/2-\epsilon$ for all $i \neq j$, which implies that $\mathcal{C}$ has relative distance at least $1/2-\epsilon$.

Since this construction has probability at least $1/2$ of producing a code with the desired rate and distance, it follows that such a code must exist. This style of argument is an example of the probabilistic method. □

Research Questions. There are several open questions relating to this topic. See (Guruswami et al., 2019, Section 8.3).

- Is the $\Omega(\epsilon^2)$ rate optimal?
- Can one explicitly describe a code achieving rate $\Omega(\epsilon^2)$?
- Is there a code with rate $\Omega(\epsilon^2)$ that can be efficiently decoded?

Recent research. Very recently, startling progress has been made on the second and third questions. A breakthrough of Ta-Shma in STOC 2017 has shown that there are explicit codes with distance $1/2-\epsilon$ and rate $\Omega(\epsilon^{2+\alpha})$ where $\alpha \to 0$ as $\epsilon \to 0$. Moreover, in STOC 2021, Jeronimo et al. showed that these codes can be decoded in nearly-linear time.

Exercises

Exercise 22.1. For this exercise, define the ***Hamming ball***, for any $x \in \{0,1\}^n$ and $d \geq 0$, to be

$$
B(x,d) \;=\; \{\, y \in \{0,1\}^n \,:\, \Delta(x,y) \leq d \,\}\,.
$$

Here $\Delta(x,y)$ is the Hamming distance, defined in (22.3.3). Assume that $d \ll n/2$. Use the Chernoff bound to prove an upper bound on the volume $|B(x,d)|$.

Exercise 22.2. The Hoeffding bound shows that a sum of n independent indicator RVs deviates from its expectation by $\sqrt{n}$ with constant probability. This question (roughly) shows that the norm of a random vector deviates from its expectation by $O(1)$ with constant probability!

Let X be uniformly distributed on $\{0,1\}^n$.

Part I. Prove that $\mathrm{E}\left[\, \|X\|_2^2 \,\right] = n/2$.

Part II. Note that $\mathrm{E}[\, \|X\|_2 \,] \neq \sqrt{\mathrm{E}\left[\, \|X\|_2^2 \,\right]}$ in general. Prove that $\mathrm{E}[\, \|X\|_2 \,] \leq \sqrt{\mathrm{E}\left[\, \|X\|_2^2 \,\right]} = \sqrt{n/2}$.

Part III. Prove that, for all $t \geq 0$,

$$\Pr\left[\, \|X\|_2 \geq \sqrt{n/2} + t \,\right] \;\leq\; \exp(-4t^2).$$

Exercise 22.3 **Sparse strategies for zero-sum games.**

Background. In a zero-sum game[3] with payoff matrix $A \in \mathbb{R}^{m \times n}$, there are two players that we will name Alice and Bob. Alice has to choose one action in $[m]$ while, simultaneously, Bob has to pick one action in $[n]$. Both Alice and Bob know that if Alice picks action $i \in [m]$ and Bob picks $j \in [n]$, then Alice will have to pay $A_{i,j}$ dollars to Bob. In such a game, Alice wants a strategy that minimizes how much money she has to pay Bob, while Bob wants a strategy that maximizes how much money he receives from Alice.

From everyday experience (e.g., rock paper scissors), we know that it often make sense to use a randomized strategy. Let

$$\Delta_n \;=\; \left\{\, p \in [0,1]^n \;:\; \textstyle\sum_{j=1}^{n} p_j = 1 \,\right\}$$

be the set of all probability distributions over these n choices. Suppose that Alice chooses $r \in \Delta_m$ then takes action $i \in [m]$ with probability r_i. Similarly, Bob chooses $p \in \Delta_n$ then takes action $j \in [n]$ with probability p_j. With these randomized strategies, one can easily verify that the expected amount of money Alice pays Bob is $r^{\mathsf{T}} A p$.

The central result of zero-sum games, due to von Neumann, proves that there are randomized strategies that are simultaneously optimal for both Alice and Bob! Formally, there exist $p^* \in \Delta_n$ and $r^* \in \Delta_m$ such that

$$\min_{r \in \Delta} \max_{p \in \Delta} r^{\mathsf{T}} A p \;=\; \max_{p \in \Delta} \min_{r \in \Delta} r^{\mathsf{T}} A p \;=\; (r^*)^{\mathsf{T}} A p^*. \tag{22.3.5}$$

Although one can efficiently compute these optimal strategies using linear programming, they may give positive probability to almost all actions, so they are *dense*.

In this exercise, we will see that there are nearly-optimal strategies that are *sparse*. More concretely, let us assume that $A_{i,j} \in [0,1]$ for all i, j. Fix some parameter $\epsilon > 0$. We will show Bob has a strategy giving positive probability to $O(\log(m)/\epsilon^2)$ actions. By symmetry, the same is true for Alice.

Henceforth, fix $p \in \Delta_n$. Let $X_1, \ldots, X_k$ be i.i.d. random variables satisfying

$$\Pr[\, X_s = j \,] \;=\; p_j \qquad \forall j \in [n].$$

This is called a categorical distribution.

[3] For more details about zero-sum games, check out this video or Wikipedia.

Part I. Show that for any $i \in [m]$ we have

$$\Pr\left[\sum_{j=1}^{n} A_{i,j} p_j - \frac{1}{k} \sum_{s=1}^{k} A_{i,X_s} > \epsilon \right] \leq e^{-2\epsilon^2 k}.$$

Part II. Define $q \in [0,1]^n$ by $q_j = k_j/k$, where $k_j = |\{\, s \in [k] \,:\, X_s = j \,\}|$. Show that

$$\Pr\left[\sum_{j=1}^{n} A_{i,j} p_j - \sum_{j=1}^{n} A_{i,j} q_j > \epsilon \text{ for some } i \in [m] \right] \leq m e^{-2\epsilon^2 k}.$$

Part III. Show that if $k = \left\lceil \frac{\ln(m+1)}{2\epsilon^2} \right\rceil$, then there is $q \in \Delta_n$ with at most k non-zero entries such that

$$\sum_{j=1}^{n} A_{i,j} p_j - \sum_{j=1}^{n} A_{i,j} q_j \leq \epsilon \qquad \forall i \in [m].$$

Part IV. Let μ^* be the optimal value given by (22.3.5). Conclude that there is a randomized strategy $q \in \Delta_n$ for Bob that puts positive probability on at most $O(\frac{\log m}{\epsilon^2})$ actions and satisfies $\min_{r \in \Delta_m} r^{\mathsf{T}} A q \geq \mu^* - \epsilon$. That is, in expectation the optimal strategy for Bob gains at most ϵ more money than the strategy given by q.

Chapter 23

Derandomization

In this chapter we discuss the topic of **derandomization** — converting a randomized algorithm into a deterministic one.

23.1 Method of Conditional Expectations

One of the simplest methods for derandomizing an algorithm is the "method of conditional expectations". In some contexts this is also called the "method of conditional probabilities".

Exhaustive search. Let us start with a simple example. Suppose X is an random variable taking values in $[k]$. Let $f : [k] \to \mathbb{R}$ be any function and suppose $\mathrm{E}\left[\, f(X) \,\right] \leq \mu$. How can we find an $x \in [k]$ such that $f(x) \leq \mu$? Well, the assumption $\mathrm{E}\left[\, f(X) \,\right] \leq \mu$ guarantees that there *exists* $x \in [k]$ with $f(x) \leq \mu$. So we can simply use **exhaustive search** to try all possible values for x in only $O(k)$ time. The same idea can also be used to find an x with $f(x) \geq \mu$.

High-dimensional exhaustive search. Now let's make the example a bit more complicated. Suppose $X_1, \ldots, X_n$ are independent random variables taking values in $[k]$. Let $f : [k]^n \to \mathbb{R}$ be any function and suppose $\mathrm{E}\left[\, f(X_1, \ldots, X_n) \,\right] \leq \mu$. How can we find a vector $(x_1, \ldots, x_n) \in [k]^n$ with $f(x_1, \ldots, x_n) \leq \mu$? We could again perform exhaustive search, but now it will take $O(k^n)$ time. In many scenarios, this may be too costly.

Method of conditional expectations. The method of conditional expectations gives a more efficient solution than high-dimensional exhaustive search, under some additional assumptions. This method dates back to a 1973 paper of Erdos and Selfridge.

The method of conditional expectation assumes that we have some additional algorithmic access to the problem. To explain this, we need to carefully define some notation that we will use related to conditional expectations. For any $i \in [n]$ and $z \in [k]^i$, define

$$\begin{aligned} &\mathrm{E}\left[\, f(X) \mid X_{\leq i} = z_{\leq i} \,\right] \\ &\quad = \mathrm{E}\left[\, f(X_1, \ldots, X_n) \mid X_1 = z_1, \ldots, X_i = z_i \,\right]. \end{aligned} \tag{23.1.1}$$

which we can expand as

$$= \sum_{x_{i+1}\in[k]} \cdots \sum_{x_n\in[k]} \Pr[\, X_{i+1} = x_{i+1} \wedge \cdots \wedge X_n = x_n \mid X_{\leq i} = z_{\leq i} \,] \, f(z_1,\ldots,z_i,x_{i+1},\ldots,x_n).$$

Here the z_j values are not random — they are the values that we condition on. The x_j values are not random either — they are used inside the conditional expectation to enumerate all possible values that the X_j random variables can take.

Usually we will conside the case where the RVs $X_1,\ldots,X_n$ are mutually independent. In this case, conditioning on the first i variables does not affect the distribution on the remaining variables. This allows for a slightly simplified formula.

$$\begin{aligned}
\mathrm{E}[\, f(X) \mid X_{\leq i} = z_{\leq i} \,] &= \mathrm{E}[\, f(z_1,\ldots,z_i,X_{i+1},\ldots,X_n) \,] \\
&= \sum_{x_{i+1}\in[k]} \cdots \sum_{x_n\in[k]} \left(\prod_{j=i+1}^{n} \Pr[\, X_j = x_j \,] \right) f(z_1,\ldots,z_i,x_{i+1},\ldots,x_n)
\end{aligned}$$

Assumption 23.1.1. We assume we have an oracle that, given any $i \in [n]$ and any $z_1,\ldots,z_i \in [k]$, can efficiently evaluate $\mathrm{E}[\, f(X) \mid X_{\leq i} = z_{\leq i} \,]$.

Theorem 23.1.2. Given such an oracle, Algorithm 23.1 will deterministically produce a point $z \in [k]^n$ satisfying $f(z) \leq \mathrm{E}[\, f(X) \,]$.

Algorithm 23.1 The method of conditional expectations for finding $z \in [k]^n$ with $f(z) \leq \mathrm{E}[\, f(X) \,]$.

```
1: for $i = 1,\ldots,n$ do
2:     Set $z_i \leftarrow 0$
3:     repeat
4:         Set $z_i \leftarrow z_i + 1$
5:     until $\mathrm{E}[\, f(X) \mid X_{\leq i} = z_{\leq i} \,] \leq \mathrm{E}[\, f(X) \mid X_{\leq i-1} = z_{\leq i-1} \,]$
6: end for
```

Proof of Theorem 23.1.2.

Correctness. The claimed bound on $f(z)$ is proven as follows. First note that, after conditionining on $X_{\leq n} = z_{\leq n}$, no randomness remains, so $X = z$. Thus

$$\begin{aligned}
f(z) &= \mathrm{E}[\, f(X) \mid X_{\leq n} = z_{\leq n} \,] \\
&\leq \mathrm{E}[\, f(X) \mid X_{\leq n-1} = z_{\leq n-1} \,] && \text{(termination condition of } n^{\text{th}} \text{ repeat loop)} \\
&\leq \mathrm{E}[\, f(X) \mid X_{\leq n-2} = z_{\leq n-1} \,] && \text{(termination condition of } (n-1)^{\text{th}} \text{ repeat loop)} \\
&\vdots \\
&\leq \mathrm{E}[\, f(X) \mid X_{\leq 1} = z_{\leq 1} \,] \\
&\leq \mathrm{E}[\, f(X) \mid X_{\leq 0} = z_{\leq 0} \,] && \text{(termination condition of first repeat loop)} \\
&= \mathrm{E}[\, f(X) \,].
\end{aligned}$$

Termination. Next we must argue that every repeat loop will eventually succeed, with z_i set to some value in $[k]$.

This argument involves the law of total expectation (Fact A.3.15): if Y is random variable with any distribution, $\mathcal{B}_1, \ldots, \mathcal{B}_k$ are events of which exactly one must occur, and all events have positive probability, then

$$\mathrm{E}[\,Y\,] \;=\; \sum_{j=1}^{k} \Pr[\,\mathcal{B}_j\,] \cdot \mathrm{E}[\,Y \mid \mathcal{B}_j\,].$$

We apply this identity to the random variable $Y = f(X)$, where the distribution conditions on the event $X_{\leq i-1} = z_{\leq i-1}$, and with the event $\mathcal{B}_j$ being the event $X_i = j$. This yields the identity

$$\begin{aligned} &\mathrm{E}[\,f(X) \mid X_{\leq i-1} = z_{\leq i-1}\,] \\ &\quad= \sum_{z_i \in [k]} \Pr[\,X_i = z_i \mid X_{\leq i-1} = z_{\leq i-1}\,] \cdot \mathrm{E}[\,f(X) \mid X_{\leq i} = z_{\leq i}\,]. \end{aligned}$$

Note that $\sum_{z_i \in [k]} \Pr[\,X_i = z_i \mid X_{\leq i-1} = z_{\leq i-1}\,] = 1$. Thus, $\mathrm{E}[\,f(X) \mid X_{\leq i-1} = z_{\leq i-1}\,]$ is a convex combination of the values $\{\, \mathrm{E}[\,f(X) \mid X_{\leq i} = z_{\leq i}\,] \,:\, z_i \in [k] \,\}$. In particular,

$$\min_{z_i \in [k]} \mathrm{E}[\,f(X) \mid X_{\leq i} = z_{\leq i}\,] \;\leq\; \mathrm{E}[\,f(X) \mid X_{\leq i-1} = z_{\leq i-1}\,].$$

Thus, the i^{th} repeat loop will eventually find a value of z_i so that the loop terminates. $\square$

Question 23.1.3. Above we remarked that, without Assumption 23.1.1, we might need to evaluate f at $O(k^n)$ points in order to find a z such that $f(z) \leq \mathrm{E}[\,f(X)\,]$.

However, under Assumption 23.1.1, what is the runtime of Algorithm 23.1? Assume that evaluating $\mathrm{E}[\,f(X) \mid X_{\leq i} = z_{\leq i}\,]$ takes $O(1)$ time.

Answer.

Each invocation of the repeat loop requires $O(k)$ time, so the total runtime is $O(kn)$.

23.1.1 Some generalizations

Let us remark that the Method of Conditional Expectations is quite flexible, and we can generalize it in many ways.

Reversing the inequality Algorithm 23.1 finds a point $z \in [k]^n$ such that $f(z) \leq \mathrm{E}[\,f(X)\,]$. Alternatively, we could find a point $z \in [k]^n$ such that $f(z) \geq \mathrm{E}[\,f(X)\,]$. To do so, we could either reverse the inequalities in Algorithm 23.1, or we could do it by a reduction: simply replace f with $-f$.

Generalized support Suppose that X_i is a discrete random variable supported on a set $\mathcal{A}_i$. Let $f : \prod_i \mathcal{A}_i \to \mathbb{R}$ be a function. The following is an easy modification of Algorithm 23.1.

Algorithm 23.2 A generalization of Algorithm 23.1 where X_i is supported on $\mathcal{A}_i$.

```
1: for $i = 1, \ldots, n$ do
2:     Set $z_i \leftarrow \operatorname{argmin}_{\zeta \in \mathcal{A}_i} \mathrm{E}[\,f(X) \mid X_{\leq i-1} = z_{\leq i-1},\, X_i = \zeta\,]$
3: end for
```

Theorem 23.1.4. Algorithm 23.2 will output a vector $z \in \prod_i \mathcal{A}_i$ satisfying $f(z) \leq \mathrm{E}[\,f(X)\,]$. Furthermore, the total runtime is $O(\sum_i |\mathcal{A}_i|)$.

23.1.2 Example: Max Cut

To illustrate the Method of Conditional Expectations, let us consider the Max Cut problem from Section 16.1. We are given a graph $G = (V, E)$ with $n = |V|$ and $m = |E|$. Recall that this algorithm generates a cut $\delta(U)$ simply by picking a set $U \subseteq V$ uniformly at random. Equivalently, for each vertex $v \in V$, the algorithm independently flips a fair coin to decide whether to put $v \in U$. We argued that $\mathrm{E}[\,|\delta(U)|\,] \geq |E|/2$.

We will use the Method of Conditional Expectations to derandomize this algorithm. Let the vertex set of the graph be $V = [n]$. Let

$$f(x_1, \ldots, x_n) = |\delta(U)| \qquad \text{where} \qquad U = \{\, i \,:\, x_i = 1 \,\}.$$

Let $X = (X_1, \ldots, X_n)$ be independent random variables where each X_i is 0 or 1 with probability 1/2. We identify the event "$X_i = 1$" with the event "vertex $i \in U$". Then $\mathrm{E}[\, f(X)\,] = \mathrm{E}[\,|\delta(U)|\,] = |E|/2$. We wish to deterministically find values $z = (z_1, \ldots, z_n) \in \{0,1\}^n$ for which $f(z) \geq \mathrm{E}[\, f(X)\,] = |E|/2$.

In order to apply the method, we must check that Assumption 23.1.1 is satisfied. That is, given $i \in [n]$ and $z \in \{0,1\}^i$, we need to efficiently evaluate

$$\mathrm{E}[\, f(X) \mid X_{\leq i} = z_{\leq i}\,].$$

What is this quantity? It is the expected number of edges cut when we have already decided which vertices amongst $\{1, \ldots, i\}$ belong to U, and the remaining vertices $\{i+1, \ldots, n\}$ are placed in U randomly (independently, with probability 1/2). This expectation is easy to compute! For any edge with both endpoints in $\{1, \ldots, i\}$ we already know whether it will be cut or not. Every other edge has probability exactly 1/2 of being cut. So we can compute that expected value in $O(m)$ time.

Armed with this approach to evaluate the conditional expectations, we can now run[1] Algorithm 23.2. There are $O(n)$ iterations, each of which evaluates $O(1)$ conditional expectations, and each of those requires $O(m)$ time. So the total runtime is $O(nm)$.

References: (Vadhan, 2012, Section 3.4.2).

23.2 Method of Pessimistic Estimators

So far we have derandomized our very simple Max Cut algorithm. Next we will consider examples involving more sophisticated probabilistic tools, such as the Chernoff bound.

Motivating example. Let $X = (X_1, \ldots, X_n)$ be independent random variables, with $X_i \in [0,1]$. Define the function f as follows:

$$f(x_1, \ldots, x_n) \;=\; \mathbf{1}_{\sum_i x_i \geq \tau} \;=\; \begin{cases} 1 & \text{if } \sum_i x_i \geq \tau \\ 0 & \text{if } \sum_i x_i < \tau \end{cases}.$$

Thus

$$\mathrm{E}[\, f(X)\,] \;=\; \Pr[\textstyle\sum_i X_i \geq \tau\,].$$

[1]We could run Algorithm 23.1, but since X_i is supported on $\{0,1\}$ instead of on $[2] = \{1,2\}$, it might be more convenient to think of Algorithm 23.2.

Can we apply the method of conditional expectations to this function f? For any $i \in [n], z \in [0,1]^i$, we need to efficiently evaluate

$$\mathrm{E}\left[\, f(X) \;\middle|\; X_{\leq i} = z_{\leq i} \,\right] \;=\; \Pr\left[\; \textstyle\sum_i X_i \geq \tau \;\;\middle|\;\; X_1 = z_1, \ldots, X_i = z_i \;\right]. \tag{23.2.1}$$

Unfortunately, computing this is not so easy. If the X_i's were i.i.d. Bernoullis then we could compute that probability by expanding it in terms of binomial coefficients. But in the non-i.i.d. case, things are more complicated — see Exercise 23.2.

Main idea. Here is the main idea of "pessimistic estimators": instead of defining f so that its expectation *equals* the probability in (23.2.1), we will define a function g whose expectation gives an **easily-computable upper-bound** on that probability. Such a function g is called a **pessimistic estimate** of that probability.

Our standard approach to tackle probabilities of the form (23.2.1) is to use the Chernoff (or Hoeffding) bounds. So we will define our pessimistic estimator g using functions that arise in those methods.

For simplicity, suppose that $X_1, \ldots, X_n \in \{0,1\}$ are independent Bernoulli random variables. The first step of the Chernoff bound (namely (21.2.4)) shows that, for any parameter $\theta > 0$,

$$\Pr\left[\textstyle\sum_j X_j \geq \tau\right] \;=\; \mathrm{E}\left[\, f(X) \,\right] \;\leq\; \mathrm{E}\left[\exp\left(-\theta\tau + \theta\textstyle\sum_{j=1}^n X_j\right)\right] \;=\; \mathrm{E}\left[\, e^{-\theta\tau} \cdot \prod_{j=1}^n e^{\theta X_j} \,\right]. \tag{23.2.2}$$

Remark 23.2.1. It is worth noting that this step holds for *any* joint distribution on the X_i's, including any non-independent or conditional distribution. This is because we have only used exponentiation and Markov's inequality, which need no assumptions on the distribution.

Definition 23.2.2. Motivated by the inequality (23.2.2), we define

$$g(x) \;=\; g(x_1, \ldots, x_n) \;=\; \exp\left(-\theta\tau + \theta\textstyle\sum_{j=1}^n x_j\right) \;=\; e^{-\theta\tau} \cdot \prod_{j=1}^n e^{\theta x_j} \qquad \forall x \in \mathbb{R}^n. \tag{23.2.3}$$

Claim 23.2.3. For any $x \in \mathbb{R}^n$, we have $f(x) \leq g(x)$.

Proof. It is easy to see that

$$\mathbf{1}_{y \geq \tau} \;=\; \left\{\begin{matrix} 0 & \quad (\text{if } y < \tau) \\ 1 & \quad (\text{if } y \geq \tau) \end{matrix}\right\} \;\leq\; \exp(-\theta\tau + \theta y) \quad \forall y \in \mathbb{R}.$$

Thus, substituting $y = \sum_{j=1}^n x_j$, we get

$$f(x) \;=\; \mathbf{1}_{\sum_{j=1}^n x_j \geq \tau} \;\leq\; \exp\left(-\theta\tau + \theta\textstyle\sum_{j=1}^n x_j\right) \;=\; g(x),$$

as required. □

Continuing from Claim 23.2.3, if we take expectations on both sides of the inequality, we obtain

$$\mathrm{E}\left[\, f(X) \,\right] \;\leq\; \mathrm{E}\left[\, g(X) \,\right], \tag{23.2.4}$$

which is exactly the statement of (23.2.2).

Let's check that the conditional expectations are easy to compute with g. (Here we will crucially use independence of the X_i random variables.) Given any numbers $i \in [n]$, $z \in \{0,1\}^i$, we have

$$\begin{aligned} \mathrm{E}\left[\, g(X) \mid X_{\leq i} = z_{\leq i} \,\right] &= e^{-\theta\tau} \cdot \prod_{j=1}^{i} e^{\theta z_j} \cdot \mathrm{E}\left[\prod_{j=i+1}^{n} e^{\theta X_j} \mid X_{\leq i} = z_{\leq i} \right] \\ &= e^{-\theta\tau} \cdot \prod_{j=1}^{i} e^{\theta z_j} \cdot \prod_{j=i+1}^{n} \mathrm{E}\left[e^{\theta X_j} \right], \end{aligned} \tag{23.2.5}$$

where this last line is due to independence.

Is this easily computable? Let us compare the conditional expectations of f and g. The conditional expectation of f, shown in (23.2.1), was difficult to compute. The conditional expectation of g, shown in (23.2.5), is much nicer because we no longer have to compute the conditional expectation of many random variables all at once: we only have to compute unconditional expectations $\mathrm{E}\left[e^{\theta X_j} \right]$, then multiply those quantities. This is much easier.

In fact, the quantity $\mathrm{E}\left[e^{\theta X_j} \right]$ is simplify the **moment generating function** of X_j, which has explicit formulas for most random variables of interest. For example, if X_j is Bernoulli with parameter p, then

$$\mathrm{E}\left[e^{\theta X_j} \right] = pe^{\theta} + (1-p).$$

23.2.1 Applying the Method of Conditional Expectations

Now let us discuss a toy example of applying the Method of Conditional Expectations to the pessimistic estimator g. Suppose that $X_1, \ldots, X_n$ are independent random variables with each $X_j \in [0,1]$, and X_j supported on a finite set $\mathcal{A}_j$. We would like to find a vector $z \in \prod_{j=1}^{n} \mathcal{A}_j$ such that

$$\sum_{j=1}^{n} z_j \leq \tau. \tag{23.2.6}$$

Obviously this is a trivial problem because the smallest possible value of that sum is obtained by setting $z_j \leftarrow \min \mathcal{A}_j$. Nevertheless, it is instructive to see how pessimistic estimators can be used for this problem.

The starting assumption. If we were applying the ordinary Method of Conditional Expectations, we would start off by assuming that

$$\mathrm{E}\left[\sum_j z_j \right] \leq \tau. \tag{23.2.7}$$

However, because we are using the Method of Pessimistic Estimators, we will make a different assumption. First, we will assume that we have some slack in the inequality (23.2.7). Specifically, we assume that we have parameters $\hat{\mu}$ and $\delta \geq 1$ such that $\hat{\mu} \geq \mathrm{E}\left[\sum_j z_j \right]$ and $\tau = (1+\delta)\hat{\mu}$.

Using this pessimistic estimator g, we obtain the following inequalities.

$$\begin{aligned} \mathrm{E}[\,f(X)\,] &\leq \mathrm{E}[\,g(X)\,] && \text{(by (23.2.4))} \qquad (23.2.8)\\ &= \mathrm{E}\left[e^{-\theta\tau} \cdot \prod_{j=1}^{n} e^{\theta X_j} \right] && \text{(definition of } g\text{)}\\ &\leq \exp(-(1+\delta)\ln(1+\delta)\hat{\mu}/4). && \qquad (23.2.9) \end{aligned}$$

These inequalities here can all be found in the proof of the Chernoff bound. In particular, the first inequality is the same as (21.2.4). The last inequality appears in equation (b) of Theorem 21.1.1 (see also Claim 21.2.4), and it requires (see (21.2.6)) that we choose $\theta = \ln(1+\delta)$.

Our key assumption is that all parameters have been chosen to satisfy the following inequality.

$$\textbf{Starting assumption:} \qquad \exp(-(1+\delta)\ln(1+\delta)\hat{\mu}/4) \ < \ 1. \qquad (23.2.10)$$

As shown in (23.2.9), this implies that

$$\mathrm{E}[\,g(X)\,] \ < \ 1.$$

Furthermore, by (23.2.8), this also implies that

$$\Pr\left[\sum_j z_j \geq \tau \right] \ = \ \mathrm{E}[\,f(X)\,] \ < \ 1.$$

Therefore, there must *exist* a vector $z \in \prod_j \mathcal{A}_j$ with $\sum_i z_i < \tau$. However our goal is to efficiently construct such a vector z, and that is where the pessimistic estimator g shows its real power.

The algorithm. We now explain how to efficiently and deterministically find such a vector. We simply apply the method of conditional expectations to g. Under the assumption (23.2.10), we know that

$$\mathrm{E}[\,g(X)\,] \ < \ 1.$$

By Theorem 23.1.4, Algorithm 23.2 will output a vector $z \in \prod_j \mathcal{A}_j$ satisfying $g(z) < 1$.

What can we conclude about z? We know that

$$\begin{aligned} \text{(by Theorem 23.1.4):} \qquad 1 &> g(z) = \exp\Big(-\theta\tau + \theta \textstyle\sum_{j=1}^{n} z_j \Big)\\ \text{(take the log):} \qquad 0 &> -\theta\tau + \theta \sum_{j=1}^{n} z_j\\ \text{(divide by } \theta \text{ and rearrange):} \qquad \tau &> \sum_{j=1}^{n} z_j \end{aligned}$$

Thus we deterministically constructed a vector $z \in \prod_j \mathcal{A}_j$ satisfying (23.2.6), which was our original goal.

23.2.2 Congestion Minimization

Now we discuss a much more substantial application of pessimistic estimators. Recall that Section 22.2 presented a randomized algorithm that gives a $O(\log n / \log\log n)$ approximation to the congestion minimization problem. We will now convert that into a deterministic algorithm via the method of pessimistic estimators.

Recall that an instance of the problem consists of a graph $G = (V, E)$ and a sequence $(s_1, t_1), \ldots, (s_k, t_k)$ of pairs of vertices. Let $n = |V|$ and $m = |E|$. We want to find s_i-t_i paths such that each edge e is contained in few paths. Let $\mathcal{P}^i$ be the set of *all* paths in G from s_i to t_i. For every path $P \in \mathcal{P}^i$, we create a variable x_P^i. The linear programming relaxation of that problem is as follows.

$$\begin{array}{lllll}
\min & C & & & \\
\text{s.t.} & \displaystyle\sum_{P \in \mathcal{P}^i} x_P^i & = 1 & & \forall i \in [k] \\
& \displaystyle\sum_{i} \sum_{\substack{P \in \mathcal{P}^i \\ e \in P}} x_P^i & \leq C & & \forall e \in E \\
& C & \geq 1 & & \\
& x_P^i & \geq 0 & & \forall i \in [k] \text{ and } P \in \mathcal{P}^i.
\end{array}$$

The algorithm of Section 22.2 generates a sequence of paths $P = (P_1, \ldots, P_k)$, where each P_i is independently chosen such that $\Pr[\, P_i = q \,] = x_q^i$ for each $q \in \mathcal{P}^i$. Then, fixing an edge $e \in E$, Lemma 22.2.5 used a Chernoff bound to analyze the probability that at least αC^* of these randomly chosen paths traverse edge e.

Pessimistic estimator for one edge. The pessimistic estimator corresponding to this event is

$$g_e(P) \;=\; g_e(P_1, \ldots, P_k) \;=\; \exp\Big(-\theta\tau + \theta \sum_{i \in [k]} \mathbf{1}_{e \in P_i} \Big).$$

Here, as in Section 22.2.1 and (21.2.6), we use the parameters

$$\tau = \alpha C^*, \qquad \alpha = \frac{24 \ln n}{\ln \ln n}, \qquad \delta = \alpha - 1, \qquad \text{and} \qquad \theta = \ln(1 + \delta).$$

We have shown in Lemma 22.2.5 that these parameters yield

$$\mathrm{E}[\, g_e(P) \,] \;\leq\; \frac{1}{n^3}.$$

Combined pessimistic estimator. Since we want to avoid excessive congestion on all edges simultaneously, we will produce a combined pessimistic estimator g by adding the functions g_e for each edge. We define

$$g(P) \;=\; \sum_{e \in E} g_e(P).$$

Then, since $|E| \leq n^2$, we immediately have

$$\mathrm{E}[\, g(P) \,] \;\leq\; n^2 \cdot \frac{1}{n^3} \;<\; 1.$$

Thus, we may apply the Method of Conditional Expectations to g. By Theorem 23.1.4, Algorithm 23.2 will output a vector $q \in \prod_{j=1}^{k} \mathcal{P}^j$ satisfying $g(q) < 1$.

What can we conclude about the paths $q = (q_1, \ldots, q_k)$? We know that

$$\text{(by Theorem 23.1.4):} \qquad 1 \;>\; g(q) \;=\; \sum_{e \in E} \exp\Big(-\theta\tau + \theta \sum_{i \in [k]} \mathbf{1}_{e \in q_i} \Big)$$

Thus, since each term of the sum is strictly positive, for *every* $e \in E$, we have $1 > g_e(q)$. That is,

$$\begin{aligned} & 1 > \exp\Big(-\theta\tau + \theta \sum_{i\in[k]} \mathbf{1}_{e\in q_i} \Big) \qquad \forall e \in E \\ \text{(take the log):}\qquad & 0 > -\theta\tau + \theta \sum_{i\in[k]} \mathbf{1}_{e\in q_i} \qquad \forall e \in E \\ \text{(divide by } \theta \text{ and rearrange):}\qquad & \tau > \sum_{i\in[k]} \mathbf{1}_{e\in q_i} \qquad \forall e \in E \end{aligned}$$

That is, for every $e \in E$, the number of paths that traverse e is less than $\tau = \alpha C^* \leq \alpha \mathrm{OPT}$. Thus we have deterministically constructed a sequence of paths $q_1, \ldots, q_k$, with each $q_i \in \mathcal{P}^i$, such that the congestion is at most $\alpha\mathrm{OPT}$.

The last remaining detail is the runtime. Exercise 23.3 argues the algorithm can be implemented in polynomial time.

23.3 Exercises

Exercise 23.1. The greedy algorithm for Max Cut is as follows. Let $V = \{v_1, \ldots, v_n\}$ be the vertices. Initially $U \leftarrow \emptyset$ and $W \leftarrow \emptyset$. For $i = 1, \ldots, n$, we do the following: if v_i has more neighbours in W than in U, we add v_i to U, otherwise we add v_i to W. At the end of the algorithm every vertex is either in U or W, so $W = V \setminus U$.

Prove that our algorithm for Max Cut using the Method of Conditional Expectations (see Section 23.1.2) is equivalent to the greedy algorithm.

Exercise 23.2. Let $X_1, \ldots, X_n$ be independent Bernoulli random variables that are not necessarily i.i.d. Let $p_i = \mathrm{E}[X_i]$. Let $S = \sum_{i=1}^n X_i$. Write an expression for $\Pr[S \geq n/2]$ in terms of the p_i parameters.

What is the runtime of the most natural algorithm to evaluate this expression? Is it polynomial time? Exponential time?

Exercise 23.3. Let $P = (P_1, \ldots, P_k)$ be the randomly chosen paths generated by the algorithm of Section 22.2. Let g_e and g be the pessimistic estimators defined in Section 23.2.2.

Part I. Argue that, for any $i \in [k]$ and any $q \in \prod_{j=1}^i \mathcal{P}^j$, in poly($n$) time we can evaluate

$$\mathrm{E}[\, g_e(P) \mid P_{\leq i} = q \,].$$

Part II. Argue that, for any $i \in [k]$ and any $q \in \prod_{j=1}^i \mathcal{P}^j$, in poly($n$) time we can evaluate

$$\mathrm{E}[\, g(P) \mid P_{\leq i} = q \,].$$

Part III. Argue that the entire algorithm for deterministically computing the paths $q_1, \ldots, q_k$ with congestion at most $\alpha\mathrm{OPT}$ can be implemented in poly(n) time.

Exercise 23.4. Let $P \subset \mathbb{R}^d$ be a set of points satisfying $\|p\| \leq 1$ for all $p \in P$. Let $m = |P|$ and $\tau = \sqrt{2\ln(m) + 1}$.

Give a deterministic algorithm to construct signs $z_1, \ldots, z_d \in \{-1, +1\}$ such that $\max_{p \in P} \sum_{i=1}^{d} p_i z_i \leq \tau$. The algorithm should run in time $\operatorname{poly}(dm)$.

Chapter 24

Dimensionality Reduction

Dimensionality reduction is the process of mapping a high dimensional dataset to a lower dimensional space, while preserving much of the important structure. In statistics and machine learning, this often refers to the process of finding a few directions in which a high dimensional random vector has maximum variance. Principal component analysis is a standard technique for that purpose.

In this chapter we consider a different sort of dimensionality reduction. Given a set of high-dimensional points, the goal is to find lower-dimensional points whose *pairwise distances* approximately match the original points. We present a technique, known as the ***random projection method*** or ***Johnson-Lindenstrauss method***, for solving this problem.

In the previous chapters, our main tool has been the Chernoff bound. Here we will not directly use the Chernoff bound, but the main proof uses very similar ideas.

24.1 Intuition

Let us begin with some three-dimensional examples. We will measure lengths using the Euclidean norm. Our notation for the length of a vector v is $\|v\| = \sqrt{\sum_i v_i^2}$.

In our first example, the dimension can be reduced while exactly preserving pairwise distances.

Example 24.1.1. Consider the points

$$x_1 = \begin{pmatrix} 0.3237 \\ -2.1870 \\ -0.3354 \end{pmatrix} \qquad x_2 = \begin{pmatrix} 0.1327 \\ -3.5215 \\ -0.7627 \end{pmatrix} \qquad x_3 = \begin{pmatrix} -1.6022 \\ -2.2986 \\ -1.4660 \end{pmatrix}$$

They have pairwise distances

$$\|x_1 - x_2\| \approx 1.4142 \qquad \|x_1 - x_3\| \approx 2.2361 \qquad \|x_2 - x_3\| \approx 2.2361$$

We now define a linear map

$$L = \begin{pmatrix} -0.7056 & -0.4820 & -0.5193 \\ 0.5146 & -0.8525 & 0.0920 \end{pmatrix}$$

Then define two-dimensional points y_i by $y_i = Lx_i$. This yields the points

$$y_1 = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \qquad y_2 = \begin{pmatrix} 2 \\ 3 \end{pmatrix} \qquad y_3 = \begin{pmatrix} 3 \\ 1 \end{pmatrix}.$$

They have pairwise distances

$$\|y_1 - y_2\| \approx \|x_1 - x_2\| \approx 1.4142 \qquad \|y_1 - y_3\| \approx \|x_1 - x_3\| \approx 2.2361 \qquad \|y_2 - y_3\| \approx \|x_2 - x_3\| \approx 2.2361.$$

This is not so impressive because the points x live in a two-dimensional subspace, so we have simply performed an orthogonal change of coordinates to obtain their two-dimensional representation.

In our second example, it is not possible to reduce the dimension while exactly preserving pairwise distances.

Example 24.1.2. Define the points

$$x_1 = \begin{pmatrix}1\\0\\0\end{pmatrix} \qquad x_2 = \begin{pmatrix}0\\1\\0\end{pmatrix} \qquad x_3 = \begin{pmatrix}0\\0\\1\end{pmatrix} \qquad x_4 = \begin{pmatrix}1\\1\\1\end{pmatrix}.$$

They have pairwise distances

$$\|x_i - x_j\| \;=\; \sqrt{2} \qquad \forall i, j.$$

Thus, they form a regular 3-simplex (or tetrahedron) in $\mathbb{R}^3$. It can be shown that there is no representation for the 3-simplex in $\mathbb{R}^1$ or $\mathbb{R}^2$. So there is no way to reduce the dimension of these points while exactly preserving the pairwise distances.

Nevertheless, in the next section we will show that we can reduce the dimension of *any* high-dimensional point set if we are allowed to *approximate* the pairwise distance.

24.2 The Johnson-Lindenstrauss Theorem

Suppose we have n points $x_1, \ldots, x_n \in \mathbb{R}^d$. We would like to find n points $y_1, \ldots, y_n \in \mathbb{R}^t$, where $t \ll d$, such that the lengths and pairwise distances of the y vectors are approximately the same as for the x vectors. We will show that this can be accomplished while taking t to be surprisingly small.

Each vector y_i will be obtained from x_i by a linear map. Let R be a random $t \times d$ ***Gaussian matrix***. This means that each entry of R is an independent standard Gaussian random variable. Gaussians are defined in Appendix B.4.3.

Theorem 24.2.1 (Johnson-Lindenstrauss 1984). Let $x_1, \ldots, x_n \in \mathbb{R}^d$ be arbitrary. Pick any $\epsilon = (0, 1)$. There exists $t = O(\log(n)/\epsilon^2)$ such that, if R is a $t \times d$ Gaussian matrix, and $y_i = Rx_i/\sqrt{t}$ then, with probability at least $1 - 1/n$,

$$\begin{aligned} (1-\epsilon)\|x_j\| \;&\leq\; \|y_j\| \;\leq\; (1+\epsilon)\|x_j\| && \forall j \\ (1-\epsilon)\left\|x_j - x_{j'}\right\| \;&\leq\; \left\|y_j - y_{j'}\right\| \;\leq\; (1+\epsilon)\left\|x_j - x_{j'}\right\| && \forall j, j'. \end{aligned} \tag{24.2.1}$$

References: (Shalev-Shwartz and Ben-David, 2014, Section 23.2), (Dubhashi and Panconesi, 2009, Section 2.5), (Vershynin, 2018, Theorem 5.3.1), (Boucheron et al., 2012, Theorem 2.13), (Roch, 2020, Section 2.4.6), (Blum et al., 2018, Section 2.7), (Wainwright, 2019, Example 2.12).

Observations.

- The linear map R is *oblivious*: it does not depend on the points $x_1, \ldots, x_n$ at all.

- A standard Gaussian is also called a $N(0,1)$ random variable, meaning that has zero mean and variable 1. Instead of defining $y_i = Rx_i/\sqrt{t}$, we could equivalently let the entries of R have distribution $N(0,1/t)$ and define $y_i = Rx_i$. (This follows from Fact B.4.6 with $d=1$ and $\sigma_1 = 1/\sqrt{t}$.)

Whereas principal component analysis is only useful when the original data points $\{x_1,\ldots,x_n\}$ are nearly low dimensional, this theorem requires *absolutely no assumption* on the original data. Also, note that the final data points $\{y_1,\ldots,y_n\}$ have no dependence on d. The original data could live in an arbitrarily high dimension. (Although one can always assume $d \leq n$ since $x_1,\ldots,x_n$ lie in their linear span, which is a Euclidean space of dimension at most n.)

24.2.1 Overview of proof

To prove the theorem, let us define the random linear map that is used to construct the points y_j. The parameter t will be chosen appropriately below. Let R is a $t \times d$ matrix whose entries are independently drawn from $N(0,1)$, i.e., Gaussians with mean 0 and variance 1. The point y_j in Theorem 24.2.1 is simply by multiplication with R and then rescaling

$$y_j \leftarrow \frac{R \cdot x_j}{\sqrt{t}}.$$

In pseudocode, we can write the algorithm as follows.

Algorithm 24.1 The Johnson-Lindenstrauss algorithm. Each input vector x_i is in $\mathbb{R}^d$.

```
1: function JL(vector $x_1,\ldots,x_n$, int $d$, float $\epsilon$)
2:     Let $t \leftarrow O(\ln(n)/\epsilon^2)$
3:     Let $R$ be a random Gaussian matrix of size $t \times d$
4:     for $i = 1,\ldots,n$
5:         Let $y_i \leftarrow R \cdot x_i/\sqrt{t}$
6:     return $y_1,\ldots,y_n$
7: end function
```

Our proof of Theorem 24.2.1 will show that this algorithm produces vectors $y_1,\ldots,y_n$ satisfying (24.2.1) with probability at least $1-1/n$. (There is a caveat: the definition of t involves a constant that we have not specified.)

The key to proving Theorem 24.2.1 is to prove the following lemma.

Lemma 24.2.2 (Distributional JL). Let R be a $t \times d$ Gaussian matrix. Let $\delta \in (0,1]$ be arbitrary. Let $t = 8\ln(2/\delta)/\epsilon^2$. Then for all vectors $v \in \mathbb{R}^d$ with $\|v\| = 1$,

$$\Pr\left[\frac{\|Rv\|}{\sqrt{t}} \notin (1-\epsilon, 1+\epsilon) \right] \leq \delta. \tag{24.2.2}$$

Proof of Theorem 24.2.1. Set $\delta = 1/n^3$. Consider the set of vectors

$$W = \{\, x_i \,:\, i = 1,\ldots,n \,\} \,\cup\, \{\, x_i - x_j \,:\, i \neq j \,\}.$$

There are at most n^2 vectors in W.

For any non-zero $w \in W$, we may apply the DJL lemma to $v = w / \|w\|$. The event that w's norm is *not* adequately preserved is

$$\mathcal{E}_w \;=\; \left\{ \frac{\|Rw\|}{\sqrt{t}} \notin [1-\epsilon, 1+\epsilon] \cdot \|w\| \right\} \;=\; \left\{ \frac{\|Rv\|}{\sqrt{t}} \notin [1-\epsilon, 1+\epsilon] \right\},$$

because $\|Rw\| = \|Rv \cdot \|w\|\| = \|w\| \cdot \|Rv\|$. The condition (24.2.1) holds if and only if no event $\mathcal{E}_w$ occurs. Thus

$$\begin{aligned}
\Pr[\,\text{condition (24.2.1) fails to hold}\,] &= \Pr\left[\bigcup_{w \in W} \mathcal{E}_w \right] \\
&\leq \sum_{w \in W} \Pr[\,\mathcal{E}_w\,] && \text{(the union bound)} \\
&\leq |W| \cdot \delta && \text{(by Lemma 24.2.2)} \\
&\leq 1/n. \quad \square
\end{aligned}$$

24.2.2 Random Dot Products

The idea of taking a random dot product is crucial to our discussion. Consider an arbitrary vector $v \in \mathbb{R}^d$. Let $g \in \mathbb{R}^d$ be a random vector where $\mathrm{E}[\,g_i\,] = 0$ for each i. We are interested in the analyzing the dot product $g^{\mathsf{T}} v$. This quantity may not seem so interesting at first glance because, by linearity of expectation,

$$\mathrm{E}\left[g^{\mathsf{T}} v \right] \;=\; \sum_{i=1}^{d} \mathrm{E}[\,g_i\,]\, v_i \;=\; 0. \tag{24.2.3}$$

So let us also introduce the notion of ***variance***, which is defined in Definition B.4.3. We require the following fact.

Fact B.4.5. Let $g_1, \ldots, g_d$ be independent random variables with finite variance. Let $\sigma_1, \ldots, \sigma_d \in \mathbb{R}$ be arbitrary. Then $\mathrm{Var}\left[\sum_{i=1}^{d} \sigma_i g_i \right] = \sum_{i=1}^{d} \sigma_i^2 \,\mathrm{Var}[\,g_i\,]$.

We will additionally assume that $\mathrm{Var}[\,g_i\,] = 1$ for each i. Then we have

$$\begin{aligned}
\mathrm{E}\left[(g^{\mathsf{T}} v)^2 \right] &= \mathrm{Var}\left[g^{\mathsf{T}} v \right] && \text{(by (24.2.3) and (B.4.2))} \\
&= \mathrm{Var}\left[\sum_{i=1}^{d} v_i g_i \right] \;=\; \sum_{i=1}^{d} v_i^2 \,\mathrm{Var}[\,g_i\,] && \text{(by Fact B.4.5)} \\
&= \|v\|^2 .
\end{aligned}$$

This shows that $(g^{\mathsf{T}} v)^2$ is an unbiased estimator for $\|v\|^2$. Thus, we can estimate $\|v\|^2$ by estimating $\mathrm{E}\left[(g^{\mathsf{T}} v)^2 \right]$. If the estimate is sufficiently good, we can then take the square root and estimate $\|v\|$.

Which random vector? So far, the discussion is valid for *any* independent RVs g_i with mean 0 and variance 1. For example, we could take g_i to be uniform on $\{-1, 1\}$.

Our subsequent analysis will be made easier by choosing g_i to have the Gaussian distribution $N(0,1)$. This is due to a key property: *the sum of Gaussians is also Gaussian.*

Fact B.4.6. Let $g_1, \ldots, g_d$ be independent random variables where g_i has distribution $N(0,1)$. Then, for any scalars $\sigma_1, \ldots, \sigma_d$, the sum $\sum_{i=1}^d \sigma_i g_i$ has distribution $N(0, \sum_{i=1}^d \sigma_i^2) = N(0, \|\sigma\|_2^2)$.

Consequently, our random dot product $g^\mathsf{T} v$ has the distribution

$$\text{Distribution of } g^\mathsf{T} v: \quad N(0, \sum_{i=1}^d v_i^2) = N(0, \|v\|^2). \tag{24.2.4}$$

Improved estimates by averaging. A common idea in randomized algorithms is to average multiple independent estimates in order to get a higher quality estimate. (See, e.g., Section 12.2.2.) Perhaps we can get a high-quality estimate of $\|v\|^2$ by picking several random vectors g, then averaging the values of $(g^\mathsf{T} v)^2$?

This idea is exactly what the Johnson-Lindenstrauss lemma does. Our matrix R is a $t \times d$ Gaussian matrix. Let r_i refer to the i^{th} row of R. Then $r_1, \ldots, r_t$ are independent Gaussian vectors, so each component of Rv is a random dot product:

$$(Rv)_i \;=\; r_i^\mathsf{T} v \qquad \forall i \in [t].$$

The squared norm of $Rv/\sqrt{t}$ is then

$$\left\| \frac{Rv}{\sqrt{t}} \right\|^2 \;=\; \sum_{i=1}^t \frac{(Rv)_i^2}{t} \;=\; \sum_{i=1}^t \frac{(r_i^\mathsf{T} v)^2}{t}, \tag{24.2.5}$$

which is the average of t independent random dot products.

24.2.3 Proof of Lemma 24.2.2

Let us rewrite (24.2.2) as follows.

$$\begin{aligned}
\Pr\left[\frac{\|Rv\|}{\sqrt{t}} \notin (1-\epsilon, 1+\epsilon) \right] &= \Pr\left[\|Rv\|^2 \notin \big((1-\epsilon)^2,\, (1+\epsilon)^2\big) \cdot t \right] && \text{(squaring both sides)} \\
&= \Pr\left[\sum_{i=1}^t (r_i^\mathsf{T} v)^2 \notin \big((1-\epsilon)^2,\, (1+\epsilon)^2\big) \cdot t \right] && \text{(by (24.2.5))} \\
&\leq \Pr\left[\sum_{i=1}^t (r_i^\mathsf{T} v)^2 \notin (1-\epsilon,\, 1+\epsilon) \cdot t \right]
\end{aligned}$$

Here we have used the simple bounds

$$(1-\epsilon)^2 \leq 1-\epsilon \quad \text{and} \quad (1+\epsilon)^2 \geq 1+\epsilon.$$

We have shown in (24.2.4) that $r_i^\mathsf{T} v$ has the distribution $N(0, \|v\|^2)$, which is $N(0,1)$ since we assume that v is a unit vector. It now turns out that $\sum_{i=1}^t (r_i^\mathsf{T} v)^2$ has a well-known distribution too. It is the sum of the *squares* of t independent standard normal random variables, which is called the chi-squared distribution with parameter t.

Question 24.2.3. What is $\mathrm{E}\left[\|Rv\|^2 \right]$?

Answer.

We have

$$\mathrm{E}\left[\,\|Rv\|^2\,\right] \;=\; \sum_{i=1}^{t} \mathrm{E}\left[\,(r_i^\mathsf{T} v)^2\,\right] \;=\; \sum_{i=1}^{t} \mathrm{Var}\left[\,r_i^\mathsf{T} v\,\right] \;=\; t,$$

since $r_i^\mathsf{T} v$ is $N(0,1)$.

To summarize, our desired inequality is

$$\Pr\left[\; \underbrace{\sum_{i=1}^{t} (r_i^\mathsf{T} v)^2}_{X} \notin (1-\epsilon,\, 1+\epsilon) \cdot \underbrace{t}_{\mathrm{E}[\,X\,]} \;\right] \;\leq\; \delta \qquad (24.2.6)$$

where X is a chi-squared RV with parameter t. Fortunately, tail bounds for these RVs are known.

Lemma 24.2.4 (Tail bound for chi-squared). Let X have the chi-squared distribution with parameter t. Then

$$\Pr\left[\,|X-t| \geq \epsilon t\,\right] \;\leq\; 2\exp(-\epsilon^2 t/8) \qquad \forall \epsilon \in (0,1).$$

References: (Shalev-Shwartz and Ben-David, 2014, Lemma B.12), (Wainwright, 2019, Example 2.11 and Example 2.28).

This lemma easily allows us to prove (24.2.6), which proves (24.2.2).

$$\Pr\left[\,X \notin (1-\epsilon,\, 1+\epsilon)\cdot t\,\right] \;=\; \Pr\left[\,|X-t| \geq \epsilon t\,\right] \;\leq\; 2\exp(-\epsilon^2 t/8) \;=\; \delta, \qquad (24.2.7)$$

since $t = 8\ln(2/\delta)/\epsilon^2$.

24.2.4 Example of a chi-squared tail bound

To illustrate one strategy for proving Lemma 24.2.4, let us prove the following bound on the right tail. This even yields the better constant in the exponent of $1/2$ rather than $1/8$.

Claim 24.2.5. Let X have the chi-squared distribution with parameter t. Then $\Pr\left[\,X \geq t(1+\epsilon)^2\,\right] \leq \exp(-t\epsilon^2/2)$.

Proof. Our proof will follow the Chernoff bound strategy. For any $\alpha \in \mathbb{R}$ and $\theta \geq 0$, we have

$$\Pr\left[\,X \geq \alpha\,\right] \;=\; \Pr\left[\,e^{\theta X} \geq e^{\theta\alpha}\,\right] \;\leq\; e^{-\theta\alpha}\,\mathrm{E}\left[\,e^{\theta X}\,\right]. \qquad (24.2.8)$$

The quantity $\mathrm{E}\left[\,e^{\theta X}\,\right]$ is called the moment generating function, and for many standard distributions it has a known closed form. We now cheat by referring to Wikipedia, where we find that the moment generating function for the chi squared distribution is $\mathrm{E}\left[\,e^{\theta X}\,\right] = (1-2\theta)^{-t/2}$ (for $\theta < 1/2$), so

$$\Pr\left[\,X > \alpha\,\right] \;\leq\; e^{-\theta\alpha}(1-2\theta)^{-t/2} \qquad (\text{if } \theta \in [0, 1/2)).$$

The next step is to plug in an appropriate choice of θ. We set $\theta = (1 - t/\alpha)/2$, giving

$$\begin{aligned}
\Pr[X \geq \alpha] &\leq e^{(t-\alpha)/2}(t/\alpha)^{-t/2} \\
&= \exp\left(\frac{t}{2}\Big(1-(1+\epsilon)^2\Big) - \frac{t}{2}\ln\Big(\frac{1}{(1+\epsilon)^2}\Big)\right) \qquad \text{(setting } \alpha = t(1+\epsilon)^2\text{)} \\
&= \exp\Big(-t\big(\epsilon + \epsilon^2/2 - \ln(1+\epsilon)\big)\Big) \\
&\leq \exp\Big(-t\big(\epsilon + \epsilon^2/2 - \epsilon\big)\Big) \qquad \text{(using } \ln(1+x) \leq x\text{; see Exercise B.2)} \\
&\leq \exp(-t\epsilon^2/2).
\end{aligned}$$

□

Question 24.2.6. Improve the bound $\log(1+\epsilon) \leq \epsilon$ to $\log(1+\epsilon) \leq \epsilon - \epsilon^2/4$ for $\epsilon \in [0,1]$.

Answer.

It follows from Fact B.3.5 by replacing $z \leftarrow x/2$ and integration.

24.2.5 Broader context

In this section we have switched from the world of discrete probability to continuous probability. This is to make our lives easier. The same theorem would be true if we picked the coordinates of r_i to be uniform in $\{+1, -1\}$ rather than Gaussian. But the analysis of the $\{+1, -1\}$ case is trickier, and most proofs analyze that case by showing that its failure probability is not much worse than in the Gaussian case. So the Gaussian case is really the central problem.

Second of all, you might be wondering where the name ***random projection method*** comes from. Earlier versions of the Johnson-Lindenstrauss theorem used a slightly different linear map. Specifically, they used the map Rv where $R^T R$ is a *projection* onto a uniformly random subspace of dimension t. (Recall that an orthogonal projection matrix is any symmetric, positive semidefinite matrix whose eigenvalues are either 0 or 1.) One advantage of that setup is its symmetry: one can argue that the failure probability in Lemma 24.2.2 would be the same if one instead chose a *fixed* subspace of dimension t and a *random* unit vector v. The latter problem can be analyzed by choosing the subspace to be the most convenient one of all: the span of the first t vectors in the standard basis.

So how is our map $R/\sqrt{t}$ different? It is almost a projection, but not quite. If we chose R to be a matrix of independent Gaussians, it turns out that the range of $R^T R/t$ is indeed a uniformly random subspace, but its eigenvalues are not necessarily in $\{0, 1\}$. If we had insisted that the random vectors r_i that we choose were *orthonormal*, then we would have obtained a projection matrix. We could explicitly orthonormalize them by the Gram-Schmidt method, but fortunately that turns out to be unnecessary: the Johnson-Lindenstrauss theorem is true, even if we ignore orthonormality of the r_i's.

Our linear map $R/\sqrt{t}$ turns out to be a bit more convenient in some algorithmic applications, because we avoid the awkward Gram-Schmidt step.

Optimality. Recently there has been much progress on understanding the optimality of these results. The DJL lemma is actually optimal, up to constant factors.

Theorem 24.2.7 (Jayram-Woodruff 2013, Kane-Meka-Nelson 2011). Any f satisfying the DJL lemma must satisfy $t = \Omega(\log(1/\delta)/\epsilon^2)$.

But, this does not necessarily imply that Theorem 24.2.1 is optimal; perhaps the theorem can be proven without using the DJL lemma. Alon proved the following lower bound. (See Theorem 9.3 of this paper.)

Theorem 24.2.8 (Alon). Let $x_1, \ldots, x_{n+1} \in \mathbb{R}^n$ be the vertices of a simplex, i.e., $\|x_i - x_j\| = 1$ for all $i \neq j$. If $y_1, \ldots, y_{n+1} \in \mathbb{R}^t$ satisfy (24.2.1), then $t = \Omega(\frac{\log(n)}{\epsilon^2 \log(1/\epsilon)})$.

This shows that Theorem 24.2.1 is almost optimal, up to the factor $\log(1/\epsilon)$ in the denominator. Actually, for this particular set of points (the vertices of a simplex), Theorem 24.2.1 is *not* optimal and Alon's bound is the right one. However, there is a different point set showing that Theorem 24.2.1 is in fact optimal.

Theorem 24.2.9 (Larsen-Nelson FOCS 2017). There exist points $x_1, \ldots, x_n \in \mathbb{R}^d$ such that the following is true. Consider any map $L : \mathbb{R}^d \to \mathbb{R}^t$, let $y_j = L(x_j)$, and suppose that (24.2.1) is satisfied. Then $t = \Omega(\log(n)/\epsilon^2)$.

Other norms and metrics. The Johnson-Lindenstrauss lemma very strongly depends on properties of the Euclidean norm. For other norms, this remarkable dimensionality reduction is not necessarily possible. For example, for the ℓ_1 norm $\|x\|_1 := \sum_i |x_i|$, it is known that any map into $\mathbb{R}^d$ that preserves pairwise ℓ_1-distances between n points up to a factor $c \geq 1$ must have $d = \Omega(n^{1/c^2})$. If $c = 1 + \epsilon$, then there are upper bounds of $d = O(n \log n/\epsilon^2)$ and $d = O(n/\epsilon^2)$.

References: Talagrand Proc. AMS 1990, Brinkman-Charikar FOCS 2003, Lee-Naor 2004, Newman-Rabinovich SODA 12.

For more on this subject, see the survey of Indyk and Matousek or the tutorial of Indyk.

24.3 Fast Johnson-Lindenstrauss

In the previous section we saw the Johnson-Lindenstrauss theorem on dimensionality reduction. Suppose we have n points in $\mathbb{R}^d$ and we map them to $\mathbb{R}^t$, where $t = O(\log(n)/\epsilon^2)$, simply by applying a $t \times d$ matrix of Gaussians (scaled appropriately). Then all lengths and pairwise distances are preserved up to a factor $1 + \epsilon$, with high probability. This is a very useful tool in algorithm design.

Let's consider the efficiency of such a mapping. Directly applying matrix-vector multiplication, the time to map a single point from $\mathbb{R}^d$ to $\mathbb{R}^t$ is $O(td)$. There has been much work on trying to make this faster.

One direction of research considered using a slightly *sparse* matrix instead of a dense matrix of Gaussians. The state of the art (Kane-Nelson JACM 2014) allows the matrix to have only an ϵ fraction of non-zero entries, so the time to map a single point becomes $O(\epsilon t d)$. Their result is optimal to with a factor of $O(\log(1/\epsilon))$.

Today we discuss a different line of research. Instead of using sparse matrices, we will use *structured* matrices, for which multiplication can be done faster than the naive algorithm. (As a trivial example, consider the matrix of all ones. It is dense, but multiplying by it is very easy.) Such matrices are called "Fast Johnson-Lindenstrauss Transforms", and they have been used extensively in the algorithms, compressed sensing, machine learning and numerical linear algebra communities.

The first result of this type is stated below. Is is of the Distributional JL type, in that it preserves the length of any fixed vector with good probability.

Theorem 24.3.1 (Ailon-Chazelle STOC 2006). There is a $t \times d$ random matrix that satisfies the DJL lemma and for which matrix-vector multiplication takes time $O(d \log d + t^3)$.

To understand if this runtime is good, consider the scenario where we are applying dimensionality reduction to n data points. The Ailon-Chazelle result is only interesting for certain parameters n, d and t. First, suppose n is very small, say $d = n$, so $t \approx \log n = \log d$. Then the original JL theorem takes time roughly $O(td) = O(d \log d)$ per vector, which is the same as Ailon-Chazelle. Second, suppose n is very large, say $n = 2^{\sqrt{d}}$, so $t \approx \log n = d^{1/2}$. Then the original JL theorem takes time $O(td) = O(d^{3/2})$ per vector and Ailon-Chazelle also takes time $O(d \log d + t^3) = O(d^{3/2})$. The Ailon-Chazelle result is interesting in the intermediate range, where $d \ll n \ll 2^{d^{1/2}}$

We will prove the following theorem, which is a slightly simplified form of the Ailon-Chazelle result.

Theorem 24.3.2. There is a random matrix R of size $t \times d$ with $t = O(\log(d/\delta)^2 \log(1/\delta)/\epsilon^2)$ such that, for each $x \in \mathbb{R}^d$,

$$\|Rx\| \in [1-\epsilon, 1+\epsilon] \cdot \|x\|$$

holds with probability at least $1-\delta$. Matrix-vector multiplication with R takes time $O(d \log d + t)$.

24.3.1 A Simple Start: Super-Sparse Sampling

Let's start with simple idea: given a vector x, we will sample it using a very sparse sampling matrix. This matrix is denoted S and it has size $t \times d$. Each row of S has a single non-zero entry of value $\sqrt{d/t}$ in a uniformly random location.

For any vector $x \in \mathbb{R}^d$, we have

$$\mathrm{E}\left[\, (Sx)_i^2 \,\right] \;=\; \sum_{j=1}^{d} \underbrace{\Pr\left[\, i^{\text{th}} \text{ row's nonzero entry is in location } j \,\right]}_{=1/d} \cdot (\sqrt{d/t})^2 \cdot x_j^2 \;=\; (1/t) \cdot \|x\|_2^2$$

$$\Longrightarrow \quad \mathrm{E}\left[\, \|Sx\|^2 \,\right] \;=\; \mathrm{E}\left[\sum_{i=1}^{t} (Sx)_i^2\right] \;=\; \|x\|_2^2 \tag{24.3.1}$$

So this works well in expectation, even if we have $t = 1$. Unfortunately the variance can be terrible.

Example 24.3.3 (Sparse case). Suppose x has just one non-zero coordinate, say $x = e_1$. The expected number of non-zero coordinates in Sx is t/d. So Sx is very likely to be 0 unless $t = \Omega(d)$. So there is very little dimensionality reduction.

Example 24.3.4 (Dense case). Suppose $x = [1, 1, \cdots, 1]/\sqrt{d}$, which has $\|x\|_2 = 1$. Then $Sx = \sqrt{1/t}x$, so $\|Sx\| = \sqrt{1/t} \cdot \sqrt{t} = 1$. Thus the norm of x is preserved exactly, regardless of the value of t.

Let us try to extract some general principles from these examples.

- *Sparse case.* If one coordinate is much larger than the others in absolute value (as in Example 24.3.3) then this approach seems unlikely to work.
- *Dense case.* If all coordinates have similar absolute value (as in Example 24.3.4, then the approach seems promising. In this case, we also say that x is "uncorrelated with the standard basis".

To make these principles mathematically precise, we turn to the language of norms. Let us recall the following standard inequality.

Fact B.2.1. For all $x \in \mathbb{R}^d$,

$$\|x\|_p \;\leq\; \|x\|_r \;\leq\; d^{1/r-1/p} \cdot \|x\|_p \qquad \forall 1 \leq r \leq p \leq \infty.$$

In particular, the most useful cases are

$$\|x\|_1 \;\geq\; \|x\|_2 \;\geq\; \|x\|_\infty$$
$$\frac{1}{\sqrt{d}} \|x\|_1 \;\leq\; \|x\|_2 \;\leq\; \sqrt{d} \cdot \|x\|_\infty .$$

Let us now consider the norms in the preceding examples.

- *Sparse case.* In Example 24.3.3, we have $\|x\|_2 = \|x\|_\infty = 1$.
- *Dense case.* In Example 24.3.3, we have $\|x\|_2 = 1$ but $\|x\|_\infty = 1/\sqrt{d}$.

Inspired by the examples, we use the ratio $\|x\|_\infty / \|x\|_2$ as a measure of sparsity of the vector x.

$$\begin{aligned} x \text{ is sparse:} \qquad & \frac{\|x\|_\infty}{\|x\|_2} \gtrapprox 1 \\ x \text{ is dense:} \qquad & \frac{\|x\|_\infty}{\|x\|_2} \lessapprox \frac{1}{\sqrt{d}}. \end{aligned}$$

By Fact B.2.1, these are the most extreme values of this ratio.

The following claim formally proves that super-sparse sampling works when x is dense. Define

$$\lambda \;=\; \sqrt{\frac{2\ln(4d/\delta)}{d}}. \tag{24.3.2}$$

Claim 24.3.5. Let y be a fixed vector in $\mathbb{R}^d$ with $\|y\|_2 = 1$ and $\|y\|_\infty \leq \lambda$. Let S be a $t \times d$ super-sparse sampling matrix with $t = 2\ln(4d/\delta)^2 \ln(4/\delta)/\epsilon^2$. Then

$$\Pr\Big[\, \|Sy\|_2^2 \notin (1-\epsilon, 1+\epsilon) \,\Big] \;\leq\; \delta/2.$$

Proof. See Exercise 24.5. □

24.3.2 Idea: Rotating the basis

Stating the problematic scenario in these terms leads to a useful idea. We said that super-sparse sampling will require large t when $\|x\|_\infty / \|x\|_2 \approx 1$. Now we recall an important property of the Euclidean norm: it is invariant under rotations and reflections. Formally, $\|Mx\|_2 = \|x\|_2$ for any orthogonal matrix M. Another way to say this is that $\|\cdot\|_2$ is invariant under an orthogonal change of basis, and indeed $\|\cdot\|_2$ can be defined without reference to any basis (using an inner product). On the other hand, the infinity norm $\|\cdot\|_\infty$ is *heavily* dependent on the choice of basis. For example,

$$\begin{aligned} u = (1,0,0,\ldots,0) \in \mathbb{R}^d \qquad & \text{has} \qquad \|u\|_2 = \|u\|_\infty = 1 \\ v = (1,1,\ldots,1)/\sqrt{d} \in \mathbb{R}^d \qquad & \text{has} \qquad \|v\|_2 = 1 \text{ and } \|v\|_\infty = 1/\sqrt{d} \end{aligned}$$

even though v is a rotation of u. Intuitively, the quantity $\|x\|_\infty / \|x\|_2$ is telling us how well the vector x "aligns" with the standard basis.

In order for our super-sparse sampling to work we need x to be dense, which means that x is not aligned with the standard basis. However we have no control over x because our theorem needs to work for all x. The key idea is to control our basis instead. Why not choose a random basis that is unlikely to align with the given vector x? Indeed, that is basically what is accomplished by the dense matrix of Gaussians in the previous section.

The only issue with the dense matrix of Gaussians is that matrix-vector multiplications are too slow. So let's think if there is a quicker way to randomly rotate the basis. Whenever I think of vectors that "disagree" with the standard basis, the first object that comes to mind is a Hadamard matrix.

Definition 24.3.6. A ***Hadamard matrix*** is a $d \times d$ real matrix H that is orthogonal (meaning $H^{\mathsf{T}} H = I$) and has all entries in $\left\{ \pm 1/\sqrt{d} \right\}$.

It is not a priori clear that Hadamard matrices exist, and indeed it is not fully known for what values of d they exist[1]. In the case $d = 2$, we have the Hadamard matrix

$$H_2 \;=\; \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} / \sqrt{2},$$

which amounts to a 45-degree rotation of the standard basis. In fact, we can build on this recursively to obtain a Hadamard matrix whenever d is a power of two.

$$H_d \;=\; \begin{pmatrix} H_{d/2} & H_{d/2} \\ H_{d/2} & -H_{d/2} \end{pmatrix} / \sqrt{2}. \tag{24.3.3}$$

This is called a Walsh-Hadamard matrix, and it has many nice properties.

First of all H_d is symmetric, so $H_d^{\mathsf{T}} = H_d$. To see that H_d is indeed a Hadamard matrix, we must make two observations. First, induction shows that every entry of H_d is $\pm(1/\sqrt{2})^{\log_2 d} = \pm 1/\sqrt{d}$. Second, we have

$$H_d^{\mathsf{T}} H_d \;=\; H_d H_d \;=\; \begin{pmatrix} H_{d/2} & H_{d/2} \\ H_{d/2} & -H_{d/2} \end{pmatrix} \cdot \begin{pmatrix} H_{d/2} & H_{d/2} \\ H_{d/2} & -H_{d/2} \end{pmatrix} / 2 \;=\; \begin{pmatrix} 2H_{d/2}H_{d/2} & 0 \\ 0 & 2H_{d/2}H_{d/2} \end{pmatrix} / 2 \;=\; I,$$

so H_d is orthogonal.

Another nice property of H_d is that matrix-vector multiplication can be performed in $O(d \log d)$ time. This follows from an easy divide-and-conquer algorithm.

24.3.3 Randomized Hadamard Matrix

Using H_d to change our basis will not guarantee that x is dense. It is obvious that, for *any* fixed orthogonal matrix M, there exist vectors x for which $\|Mx\|_\infty / \|Mx\|_2 = 1$: simply let x be any row of M. This is why the randomization is necessary.

Instead, we must pick a *random* change of basis M and argue that

$$\text{each } x \in \mathbb{R}^d \text{ satisfies} \qquad \left(\; \frac{\|Mx\|_\infty}{\|Mx\|_2} \approx \frac{1}{\sqrt{d}} \quad \text{with high probability} \; \right). \tag{24.3.4}$$

So far our Hadamard matrix $H = H_d$ has no randomness. How can we "randomize" it? Well, in the recursive definition (24.3.3) we quite arbitrarily put the minus sign in the lower-right quadrant. The construction works just as well if we put the minus sign in any quadrant, so this suggests that we should be able to randomize the construction using random signs.

We will introduce random signs in a slightly more convenient way. Let D be a diagonal matrix whose i^{th} diagonal entry is a random sign $\xi_i \in \{-1, 1\}$, and these are independent. Our random Hadamard matrix is the product $M = HD$. This is indeed a Hadamard matrix: its entries are still $\pm 1/\sqrt{d}$, and M is orthogonal because $M^{\mathsf{T}} M = DH^{\mathsf{T}} HD = D^2 = I$.

The following claim shows that, with this randomized Hadamard matrix M, every vector x satisfies (24.3.4).

[1] One open question is that a $d \times d$ Hadamard matrix exists whenever d is a multiple of 4.

Claim 24.3.7. Let $x \in \mathbb{R}^d$ be non-zero. Let $y = HDx$, where HD is the random Hadamard matrix. Then

$$\Pr_D\left[\frac{\|y\|_\infty}{\|y\|_2} \geq \underbrace{\sqrt{\frac{2\ln(4d/\delta)}{d}}}_{\lambda} \right] \;\leq\; \delta/2.$$

Proof. It suffices to consider the case $\|x\|_2 = 1$, because $\|HDx\|_\infty / \|HDx\|_2$ is invariant under rescaling x. Note that $\|HDx\|_2 = \|x\|_2 = 1$ as HD is orthogonal, so it suffices to show that all coordinates of HDx are likely small. We will follow our usual approach of showing that each coordinate is very likely to be small, then union bounding over all coordinates.

Consider y_1, the first coordinate of $y = HDx$. It is obtained by multiplying each coordinate of x by a random sign, then taking the dot-product with the first row of H. So

$$y_1 \;=\; \sum_j \xi_j H_{1,j} x_j, \tag{24.3.5}$$

Note that the terms of this sum are independent random variables. It is tempting to apply the Chernoff bound to analyze y_1, but the Chernoff bound that we have used so far is only valid for sums of non-negative random variables. Instead, we will use the generalized form of the Hoeffding bound, Theorem 21.3.5.

We apply that theorem with $X_j = \xi_j H_{1,j} x_j$. Since $\xi_j \in \{-1,+1\}$, we have $X_j = \pm H_{1,j} x_j$. So X_j lies in the interval $[a_j, b_j]$ where $-a_j = b_j = |H_{1,j} x_j|$. Note that $\mathrm{E}[\,X_j\,] = 0$ and

$$\sum_{j=1}^{d} (b_j - a_j)^2 \;=\; 4\sum_{j=1}^{d} H_{1,j}^2 x_j^2 \;=\; 4\sum_{j=1}^{d} (1/d) x_j^2 \;=\; 4\,\|x\|_2^2/d \;=\; 4/d.$$

Defining $q = \delta/2d$, the quantity s in Theorem 21.3.5 becomes $s = \sqrt{2\ln(4d/\delta)/d}$, which equals the value of λ defined in (24.3.2). Consequently, (24.3.5) and Theorem 21.3.5 yield

$$\Pr[\,|y_1| \geq \lambda\,] \;=\; \Pr\left[\left| \textstyle\sum_j X_j - \underbrace{\mathrm{E}\left[\textstyle\sum_j X_j\right]}_{=0} \right| \geq s \right] \;\leq\; \delta/2d.$$

A union bound over all coordinates of y shows that

$$\Pr[\,\|y\|_\infty \geq \lambda\,] \;\leq\; \sum_{j=1}^{d} \Pr[\,|y_j| \geq \lambda\,] \;\leq\; d \cdot (\delta/2d) \;=\; \delta/2. \qquad \square$$

24.3.4 Putting it all together

Earlier we said that super-sparse sampling should work as long as x is dense holds. We have just shown x is likely to be dense after applying the randomized Hadamard matrix; more precisely, (24.3.4) holds with $M = HD$. The final step is to apply the super-sparse sampling matrix S to Mx. To summarize, the overall linear map is $R = SHD$ where S is a super-sparse sampling matrix of size $t \times d$, H is a $d \times d$ Hadamard matrix, and D is a diagonal matrix of random signs.

Proof of Theorem 24.3.2. Fix any vector x with $\|x\|_2 = 1$. Construct the random matrix $R = SHD$ as explained above and let $y = HDx$. Define the events

$$\begin{aligned}\mathcal{E}_1 &= \{\, \|y\|_\infty \geq \lambda \,\} \\ \mathcal{E}_2 &= \{\, \|Sy\|_2 \notin (1-\epsilon, 1+\epsilon) \,\}\end{aligned}$$

We have the following bounds.

$$\begin{aligned}\Pr[\mathcal{E}_1] &\leq \delta/2 \qquad \text{(by Claim 24.3.7)} \\ \Pr\left[\mathcal{E}_2 \mid \overline{\mathcal{E}_1}\right] &\leq \delta/2 \qquad \text{(by Claim 24.3.5).}\end{aligned}$$

Thus, by Exercise A.4, we obtain

$$\Pr[\|Rx\| \notin (1-\epsilon, 1+\epsilon)] = \Pr[\mathcal{E}_2] \leq \Pr[\mathcal{E}_1] + \Pr\left[\mathcal{E}_2 \mid \overline{\mathcal{E}_1}\right] \leq \delta.$$

Finally, let us check that matrix-vector multiplication with $R = SHD$ is efficient.

- Multiplication by D is trivial and takes $O(d)$ time.
- Multiplication by H requires $O(d \log d)$ time as explained above.
- Multiplication by S is trivial and takes $O(t)$ time.

So the total time is $O(d \log d + t)$. □

24.4 Subspace Embeddings

The Johnson-Lindenstrauss Theorem shows how to reduce the dimension while preserving distances between a finite set of points. Now we will consider how to reduce the dimension for the *infinitely many* points in a subspace.

Let U be a subspace of $\mathbb{R}^d$ with dimension k. We would like to find a matrix R of size $t \times d$ such that $t \approx k$ and

$$\frac{\|Rx\|}{\|x\|} \in [1-\epsilon, 1+\epsilon] \qquad \forall x \in U,\ x \neq 0. \tag{24.4.1}$$

Question 24.4.1. Is this trivial?

Answer.

Yes. Let $t = k$ and let the rows of R be an orthonormal basis of U. This satisfies (24.4.1) with $\epsilon = 0$.

What makes the problem non-trivial is that we want R to be *oblivious to* U, just like Johnson-Lindestrauss is oblivious to the points whose distance are preserved.

Theorem 24.4.2. For any integer k and any $\epsilon \in (0,1)$, let $t = O(k \log(1/\epsilon)/\epsilon^2)$. Let R be a $t \times d$ matrix where the entries are independent with distribution $N(0, 1/t)$. Then, for any subspace U of dimension k, we have

$$\Pr\left[\frac{\|Rx\|}{\|x\|} \in [1-\epsilon, 1+\epsilon] \ \ \forall x \in U,\ x \neq 0\right] \geq 1 - 2e^{-k}.$$

To prove the JL Theorem, we proved the DJL Lemma for individual vectors x, then extended that to a finite collection of vectors by taking a union bound. Now Theorem 24.4.2 requires that we approximate the norm for all vectors in U simultaneously. Since U is infinitely large, a naive union bound will not suffice.

A simplifying reduction. Instead of considering an arbitrary subspace U, it suffices to consider the case that $U = \operatorname{span}\{e_1, \ldots, e_k\}$. The reason stems from the following fact.

Fact 24.4.3. Let R be a $t \times d$ random Gaussian matrix and let A be any $d \times d$ orthogonal matrix. Then RA has the same distribution as R.

References: See (Vershynin, 2018, Proposition 3.3.2).

Imagine letting the first k columns of A be an orthonormal basis of U, then extending that to an orthonormal basis of $\mathbb{R}^d$. Then (24.4.1) is equivalent to

$$\frac{\|RAx\|}{\|x\|} \;\in\; [1-\epsilon, 1+\epsilon] \qquad \forall x \in \operatorname{span}\{e_1, \ldots, e_k\}. \tag{24.4.2}$$

Since RA and R have the same distribution (by Fact 24.4.3), our goal is equivalent to showing

$$\Pr\left[\frac{\|Rx\|}{\|x\|} \in [1-\epsilon, 1+\epsilon] \;\; \forall x \in \mathbb{R}^k,\, x \neq 0 \right] \;\geq\; 1 - e^{-k} \tag{24.4.3}$$

where R has now shrunk from size $t \times d$ to $t \times k$, since we only cared about $\operatorname{span}\{e_1, \ldots, e_k\}$.

24.4.1 First approach: Appling a standard theorem

The event in (24.4.3) uses important quantities from linear algebra. For any $t \times k$ matrix R, define

$$\begin{aligned} \text{Maximum singular value:} \quad s_1(R) \;&=\; \max_{x \in \mathbb{R}^k,\, x \neq 0} \frac{\|Rx\|}{\|x\|} \\ \text{Minimum singular value:} \quad s_k(R) \;&=\; \min_{x \in \mathbb{R}^k,\, x \neq 0} \frac{\|Rx\|}{\|x\|}. \end{aligned}$$

References: Wikipedia.

Using these definitions, (24.4.3) is equivalent to showing that

$$\Pr[\; 1-\epsilon \leq s_k(R) \;\wedge\; s_1(R) \leq 1+\epsilon \;] \;\geq\; 1 - 2e^{-k}. \tag{24.4.4}$$

The singular values of Gaussian matrices are very well-studied, and the following bound is known.

Theorem 24.4.4. If R has size $t \times k$ where the entries are independent $N(0, 1/t)$ then, for all $z > 0$,

$$\Pr\left[1 - \frac{\sqrt{k}+z}{\sqrt{t}} \;\leq\; s_k(R) \;\wedge\; s_1(R) \;\leq\; 1 + \frac{\sqrt{k}+z}{\sqrt{t}} \right] \;\geq\; 1 - 2\exp(-z^2/2).$$

References: See Theorem 2.13 in Davidson-Szarek, (Wainwright, 2019, Example 6.2 and Exercise 5.14), (Vershynin, 2018, Theorem 4.6.1).

Taking $t = k/\epsilon^2$ and $z = \sqrt{k}$ this proves (24.4.4) (up to constant factors) which proves Theorem 24.4.2.

24.4.2 Second approach: A direct argument

Next we observe that the function $x \mapsto \|Rx\| / \|x\|$ is a continuous function on non-zero x. Thus if $\|Rx\| / \|x\| \in [1-\epsilon, 1+\epsilon]$ holds for some x, then it approximately holds for all nearby x. Also, since $\|Rx\| / \|x\|$ only depends on the direction of x, not its norm, it suffices to consider vectors with $\|x\| = 1$.

Define the Euclidean sphere

$$S = \left\{ \, x \in \mathbb{R}^k \; : \; \|x\| = 1 \, \right\}.$$

Since we only need to consider points $y \in S$, it seems conceivable that we could find *finitely* many points $P = \{p_1, \ldots, p_\ell\}$ such that *every* point in S is "nearby" to some p_i.

Definition 24.4.5. An ***ϵ-net*** of a set S is a set $P \subseteq S$ satisfying $\min_{p \in P} \|p - x\| \leq \epsilon$ for all $x \in S$.

References: (Vershynin, 2018, Definition 4.2.1), (Wainwright, 2019, Definition 5.1), Wikipedia.

The following fact is known.

Fact 24.4.6 (ϵ-net of sphere). The sphere S in $\mathbb{R}^k$ has an ϵ-net of size $(3/\epsilon)^k$.

References: (Vershynin, 2018, Corollary 4.2.13), (Wainwright, 2019, Example 5.8).

Let P be an ϵ-net of size $(3/\epsilon)^k = \exp\big(k \ln(3/\epsilon)\big)$. Then we apply the Johnson-Lindenstrauss lemma with error parameter ϵ to the points in P. Note that $\|p\| = 1$ for all $p \in P$, since $P \subseteq S$. By (24.2.7) and a union bound, we have

$$\|Rp\| \;\in\; [1-\epsilon, 1+\epsilon] \qquad \forall p \in P \tag{24.4.5}$$

with probability at least

$$1 - |P| \cdot \exp(-\epsilon^2 t/8) \;\geq\; 1 - \exp\big(k \ln(3/\epsilon) - \epsilon^2 t/8\big).$$

This probability is at least $1 - e^{-k}$ if $t = 16k \ln(3/\epsilon)/\epsilon^2$.

Now we show that, if the norm is (approximately) preserved on the ϵ-net, then it is preserved on the entire sphere.

Claim 24.4.7. Suppose that (24.4.5) holds. Then $\|Rx\| \leq 1 + O(\epsilon)$ for all $x \in S$.

References: (Vershynin, 2018, Lemma 4.4.1).

Proof. Consider any $x^* \in \arg\max_{x \in S} \|Rx\|$. The existence of x^* comes from the Weierstrass extreme value theorem, since $x \mapsto \|Rx\|$ is continuous and S is a closed, bounded set.

Since P is an ϵ-net, there exists $p^* \in P$ such that $\|x^* - p^*\| \leq \epsilon$. By (24.4.5), we have $\|Rp^*\| \in [1-\epsilon, 1+\epsilon]$. We now bound $\|Rx^*\|$ by relating it to $\|Rp^*\|$.

$$\begin{aligned}
\|Rx^*\| &\leq \|Rp^*\| + \|R(x^* - p^*)\| && \text{(by the triangle inequality)} \\
&\leq (1+\epsilon) + \|R(x^* - p^*)\| && \text{(by (24.4.5))} \\
&= (1+\epsilon) + \left\| R \frac{(x^* - p^*)}{\|x^* - p^*\|} \right\| \cdot \|x^* - p^*\| && \text{(normalizing the vector)} \\
&\leq (1+\epsilon) + \|Rx^*\| \cdot \epsilon && \text{(since } x^* \text{ is a maximizer).}
\end{aligned}$$

Rearranging, we get $\|Rx^*\| \leq \frac{1+\epsilon}{1-\epsilon}$. Since $\frac{1}{1-\epsilon} \leq 1 + 2\epsilon$ (see Fact B.3.5), the result follows. $\square$

A similar argument yields the following claim, which proves Theorem 24.4.2.

Claim 24.4.8. Suppose that (24.4.5) holds. Then $\|Rx\| \geq 1 - O(\epsilon)$ for all $x \in S$.

24.4.3 Subspace Embeddings with Fast JL

In the preceding discussion we have assumed that R is a Gaussian matrix, which simplified matters due to the rotational invariance (Fact 24.4.3), but unfortunately multiplication by R is slow. We now discuss how to replace R with a Fast JL matrix.

The desired conclusion of Theorem 24.4.2 is that

$$\Pr\left[\frac{\|Rx\|}{\|x\|} \in [1-\epsilon, 1+\epsilon] \quad \forall x \in U,\, x \neq 0\right] \geq 1 - 2e^{-k}.$$

Now let A be a $d \times k$ matrix whose columns are an orthonormal basis of U. Recall that P is an ϵ-net for S, the sphere in $\mathbb{R}^k$. Since A is an ***isometry*** (a distance-preserving bijective map) between $\mathbb{R}^k$ and U,

- $A \cdot S = \{\, Ax \,:\, x \in S \,\}$ is the sphere in U.
- $A \cdot P = \{\, Ap \,:\, p \in P \,\}$ is an ϵ-net for $A \cdot S$.

Now apply Theorem 24.3.2 with $\delta = e^{-k}/|P| \geq (\epsilon/3e)^k$ and $t = O(\log(d/\delta)^2 \log(1/\delta)/\epsilon^2)$. It follows that

$$\Pr\left[\, \|Rx\| \in [1-\epsilon, 1+\epsilon] \;\forall x \in A \cdot P \,\right] \;\geq\; 1 - 2e^{-k}.$$

An argument similar to Claim 24.4.7 implies that

$$\begin{aligned} &\Pr\left[\, \|Rx\| \in [1-O(\epsilon), 1+O(\epsilon)] \;\forall x \in U \,\right] \\ =\; &\Pr\left[\, \|Rx\| \in [1-O(\epsilon), 1+O(\epsilon)] \;\forall x \in A \cdot S \,\right] \;\geq\; 1 - 2e^{-k}. \end{aligned} \tag{24.4.6}$$

This proves Theorem 24.4.2 with the weaker dimension of $t = O\big((k \log(d)/\epsilon)^3\big)$, but with a matrix R supporting multiplication in time $O(d \log d + t)$.

Discussion. See Section 2.1 of this survey, Section 8.7 of this survey or Section 6 of this survey.

24.5 Exercises

Exercise 24.1. Let R be a JL matrix of size $t \times d$, where the entries are independent $N(0, 1/t)$. Prove that $\mathrm{E}\left[R^{\mathsf{T}} R \right] = I$. This is similar to the notion of being in isotropic position.

Exercise 24.2. Let $x_1, \ldots, x_n$ be given vectors in $\mathbb{R}^d$, and let $\epsilon \in (0, 1)$ be arbitrary. For some $t = O(\log(n)/\epsilon^2)$, define R to be a $t \times d$ matrix whose entries are independent $N(0, 1/t)$.

Part I. Suppose that $\|x_i\| \leq 1$ for all $i \in [n]$. Prove that, with probability at least $1 - 1/n$,

$$|x_i^{\mathsf{T}} x_j - x_i R^{\mathsf{T}} R x_j| \;\leq\; \epsilon \qquad \forall i, j.$$

Part II. Now assume no constraint on the norm of $x_1, \ldots, x_n$. Prove that, with probability at least $1 - 1/n$,

$$|x_i^{\mathsf{T}} x_j - x_i R^{\mathsf{T}} R x_j| \;\leq\; \epsilon \, \|x_i\| \, \|x_j\| \qquad \forall i, j.$$

Exercise 24.3. Let R be a random Fast JL matrix, as described in Section 24.3, with arbitrary $t, d > 1$.

Part I. Is $\mathrm{E}\left[R^{\mathsf{T}} R \right] = I$? This is similar to the notion of being in isotropic position.

Part II. Is R rotationally invariant? That is, for every $d \times d$ orthogonal matrix U, do R and RU have the same distribution? Prove why or why not.

Exercise 24.4 Fast JL as a Data Structure. The Fast Johnson Lindenstrauss Transform is just a random $t \times d$ R matrix of a special form. Explain how to implement it as a data structure for which:

- it uses $O(d + \mathrm{polylog}(d/\delta)/\epsilon^2)$ bits of space,
- the i, jth entry of that matrix can be computed in $O(\log d)$ time.

Here $\mathrm{polylog}(\cdot)$ means $(\log(\cdot))^c$ for some constant c.

Exercise 24.5 Super-Sparse Sampling works. Prove Claim 24.3.5.

Hints:

- $\|Sy\|_2^2 = \sum_i (Sy)_i^2$, and these summands are independent.
- The expectation was already analyzed in (24.3.1).
- The Generalized Hoeffding bound (Theorem 21.3.5) is recommended.

Exercise 24.6 $\frac{n}{4}$-net for the Hamming Cube. Recall that Section 24.4 defined an ϵ-net with respect to the Euclidean norm for the sphere. In this exercise we will construct a $\frac{n}{4}$-net with respect to the Hamming distance for the Hamming cube $\{0, 1\}^n$. (Recall that Hamming distance was defined in (22.3.3).)

For this exercise, define the ***Hamming ball***, for any $x \in \{0,1\}$ and $d \geq 0$, to be

$$B(x,d) \;=\; \{\, y \in \{0,1\}^n \;:\; \Delta(x,y) \leq d \,\}.$$

Part I. Fix some $z \in \{0,1\}^n$ and $d > 0$. Let $x_1, \cdots, x_M$ be points sampled independently and uniformly from $\{0,1\}^n$. Suppose that $\Pr[\, z \in B(x_i, d)\,] \geq p$ for some $p \in [0,1]$. Show that

$$\Pr\left[z \notin \bigcup_{i \in [M]} B(x_i, d) \right] \;\leq\; e^{-pM}.$$

Part II. Fix $z \in \{0,1\}^n$ and let x be a point uniformly sampled from $\{0,1\}^n$. Show that

$$\Pr\left[\, z \in B(x, \tfrac{n}{4}) \,\right] \;\geq\; 2^{-n/2}.$$

Hint: You will need some bounds on binomial coefficients.

Part III. Show that there is a $\frac{n}{4}$-net for $\{0,1\}^n$ of size $O(n \cdot 2^{n/2})$.

Chapter 25

Applications of Johnson-Lindenstrauss

In this chapter we will see several applications of the Johnson-Lindenstrauss lemma. These examples illustrate two key reasons why Johnson-Lindenstrauss is so useful.

- First, it's *oblivious* to the points being transformed. We can pick the matrix ahead of time, and nevertheless it is likely to work well on whatever points it is applied.
- If there is an algorithm that involves the Euclidean norm, and whose performance is exponential in the dimension, dimensionality reduction will improve its performance to polynomial in the dimension!

25.1 Streaming algorithms for ℓ_2

Recall the streaming model from Chapter 13. We will change the notation slightly. The algorithm receives a sequence of items $(a_1, a_2, \ldots, a_n)$, where each $a_i \in [d]$. The frequency vector $x \in \mathbb{Z}^d$ is

$$x_j \;=\; |\{\, i \,:\, a_i = j \,\}| \;=\; \text{number of occurrences of } j \text{ in the stream.}$$

The objective is to estimate some properties of x while using $O(\log(dn))$ space. Properties of interest include $\|x\|_p$, or the number of non-zero entries, etc.

Today we will give a simple algorithm to estimate $\|x\|_2$. As an example of a scenario where this would be useful, consider a database table $((a_1, b_1), \ldots, (a_n, b_n))$. A self-join with the predicate $a = a$ would output all triples (a, b, b') where (a, b) and (a, b') belong to the table. What is the size of this self-join? It is simply $\|x\|^2$, where x is the frequency vector for the a values in the table. So a streaming algorithm for estimating $\|x\|$ could be quite useful in database query optimization.

The Algorithm. The idea is very simple: instead of storing x explicitly, we will store a *dimensionality reduced* form of x. Let R be a random Gaussian matrix of size $t \times d$, divided by $\sqrt{t}$. (In other words, each entry is drawn from the distribution $N(0, 1/t)$. This is a rescaling of the linear map R defined in Section 24.2.) Instead of explictly maintaining the vector x, the algorithm maintains the vector $y = R \cdot x$.

Algorithm 25.1 Estimating the ℓ_2 norm of the frequency vector.

```
function EstimateL2(int $n$)
    Set $t = O(1/\epsilon^2)$
    Let $R$ be a $t \times d$ matrix with independent entries drawn from $N(0, 1/t)$
    Let $y$ be the zero vector in $\mathbb{R}^t$
    for $i = 1, \ldots, m$ do
        Receive item $a_i$ from the stream
        Add column $a_i$ of $R$ to $y$
    end for
    return $\|y\|$
end function
```

At time step i, the algorithm receives the index $j = a_i$. This implicitly causes x_j to increase by 1. Since $y = R \cdot x$, the corresponding change in y is to add the j^{th} column of R to y.

To analyze this algorithm, we use the Johnson-Lindenstrauss lemma. In Eq. (24.2.7), we proved that

$$\Pr\left[\,(1-\epsilon)\|x\| \;\leq\; \|y\| \;\leq\; (1+\epsilon)\|x\|\,\right] \;\geq\; 1 - \exp(-\epsilon^2 t/8).$$

Thus, setting $t = \Theta(1/\epsilon^2)$, we conclude that $\|y\|$ gives a $(1+\epsilon)$ approximation of $\|x\|$ with constant probability. Alternatively, if we want y to give an accurate estimate at *each* of the n time steps, we can take $t = \Theta(\log(n)/\epsilon^2)$.

How much space does this algorithm take? The algorithm stores two objects: the vector y and the matrix R. The vector y uses $t = O(1/\epsilon^2)$ words of space, which is consistent with our goal of using very little space.

Unfortunately, the matrix R uses td words of space. Storing R explicitly is worse than storing the frequency vector x. (There is also the issue of how many bits of accuracy are needed when generating the Gaussian random variables, but we will ignore that.) However, it seems that there may be some benefits to storing R instead of x, because R contains purely random numbers. The values of R do not depend on values appearing in the stream.

- At first glance, it may seem that R needn't be stored at all: every time an entry of R is accessed, a new independent random variable could be generated. But this does not work: each time an item j appears in the stream, we must add the j^{th} column of R to y, and it must be *the same column* each time.

- Does it help to use the Fast JL transform $R = SHD$ introduced in Section 24.3? Storing this matrix R requires only $O(d+t)$ words of space, which is an improvement over the ordinary JL matrix, but still too much.

- In a practical implementation, R will be generated by a pseudorandom generator initialized by some seed. This has the advantage that we can regenerate columns of R at will by resetting the seed. This is likely to work well in practice, but may not have provable guarantees.

- There is another approach that has been studied by theorists, and has provable guarantees but is probably too complicated to use in practice. Theorists have developed provably good pseudo-random generators, but only for algorithms *that use a small amount of space*. Since streaming algorithms do indeed use little space, this approach can indeed be used to regenerate the matrix R as necessary.

References: See this paper of Nisan.

- Another approach with theoretical guarantees is to generate R using special low-independence hash functions. A basic form of these ideas was discussed in Chapter 14. We may return to this in future chapters. See (Blum et al., 2018, Section 6.2.3) or Chapter 6 of these notes.

25.2 Euclidean nearest neighbor

The nearest neighbor problem is a classic problem involving high-dimensional data. Given points $P = \{p_1, ..., p_n\} \in \mathbb{R}^d$, the goal is to build a (static) data structure so that, given a query point $q \in \mathbb{R}^d$, we can quickly find i minimizing $\|q - p_i\|$. We focus on the Euclidean norm $\|\cdot\| = \|\cdot\|_2$, but this problem is interesting for many norms.

Trivial solutions. This problem can trivially be solved in polynomial time. We could do no processing of P, then for each query find the closest point by exhaustive search. This requires time $O(nd)$ for each query. An alternative approach is to use a k-d tree, which is a well-known data structure for representing geometric points. Unfortunately this could take $O(dn^{1-1/d})$ time for each query, which is only a substantial improvement over exhaustive search when the dimension d is a constant. This phenomenon, the failure of low dimensional methods when applied in high dimensions, is known as the "curse of dimensionality".

Overcoming the curse. We will show that the curse can be overcome through the use of randomization and approximation. The ***ϵ-approximate nearest neighbor*** problem (ϵ-NN) is defined as follows. Given a query point $q \in \mathbb{R}^d$, define

$$\mathrm{OPT} \;=\; \min_{p \in P} \|p - q\|$$

we must find a point $\hat{p} \in P$ such that

$$\|\hat{p} - q\| \;\leq\; \big(1 + O(\epsilon)\big) \cdot \mathrm{OPT}. \tag{25.2.1}$$

Our goal is to preprocess P and produce a data structure of size $\mathrm{poly}(n)$. Given a query point q, we wish to spend time say $O(d\log(n)/\epsilon^2)$ finding a point $p \in P$ satisfying (25.2.1). Imagine a setting where $n \approx 10^{10}$ is the number of web pages, $d \approx 10^3$ is the number of features describing those pages, and $\epsilon = 10^{-1}$ is the desired approximation. Our approach can answer a query in time $d\log(n)/\epsilon^2 \approx 10^6$, whereas the trivial approach requires time $nd \approx 10^{13}$.

The high-level idea of our solution is very simple. First, design an exhaustive search algorithm that requires space roughly $2^{O(d)}$ to solve ϵ-NN for d-dimensional data. Then, apply dimensionality reduction to reduce the data set to dimension $t \approx \log n$. Running the exhaustive search algorithm on the t-dimensional data only requires space $2^{O(t)} = n^{O(1)}$.

25.2.1 Point Location in Equal Balls

The first step is to reduce our problem to a simpler one, in which a query only needs to determine whether the closest point is at distance less than or greater than roughly r. This simpler problem is called the ***ϵ-Point Location in Equal Balls*** problem (ϵ-PLEB). It is defined as follows.

The input data is a collection of n points $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^d$. The notation $B(p, r)$, defined in (B.2.2), denotes the Euclidean ball of radius r around p. We will call $B(p, r)$ a small ball, and $B(p, (1+\epsilon)r)$ a big ball.

Given a query point $q \in \mathbb{R}^d$, the requirements are as follows:

- *Case 1:* $(1+\epsilon)r < \text{OPT}$.
  In this case, q is not in any big ball, i.e., $\nexists p_i$ with $q \in B(p_i, (1+\epsilon)r)$. We must output No.
- *Case 2:* $r < \text{OPT} \leq (1+\epsilon)r$.
  In this case, q is in a big ball but not any small ball. We can either output No, or output Yes together with a point p_j with $q \in B(p_j, (1+\epsilon)r)$.
- *Case 3:* $\text{OPT} \leq r$.
  In this case, q is in a small ball (i.e., $\exists p_i \in P$ with $q \in B(p_i, r)$). We must output Yes and a point p_j with $q \in B(p_j, (1+\epsilon)r)$.

Although there is some uncertainty about what happens in Case 2, we can draw some conclusions from the output.

- If the output is Yes, then we are always provided with a point p_j such that $q \in B(p_j, (1+\epsilon)r)$.
- If the output is No, then either Case 1 or Case 2 could apply. In either case,

$$\text{there is no } p \in P \text{ s.t. } q \in B(p, r). \tag{25.2.2}$$

25.2.2 Reduction of ϵ-NN to ϵ-PLEB

We now explain how to solve the ϵ-NN problem using a solution to the ϵ-PLEB problem. First scale the point set P so that the minimum interpoint distance is at least 1, then let Δ be the maximum interpoint distance. So $1 \leq \|p - p'\| \leq \Delta$ for all $p, p' \in P$.

Initialization. To initialize the data structure, we create an instance of ϵ-PLEB(r) for every radius r in

$$\mathcal{R} \;=\; \left\{(1+\epsilon)^0, (1+\epsilon)^1, (1+\epsilon)^2, \ldots, \Delta\right\}. \tag{25.2.3}$$

Question 25.2.1. How many different radii are there?

Answer.

There are

$$\log_{1+\epsilon}(\Delta) \;=\; \frac{\ln(\Delta)}{\ln(1+\epsilon)} \;\leq\; O(\log(\Delta)/\epsilon),$$

by Exercise B.2.

Queries. Fix a query point $q \in \mathbb{R}^d$. Let us consider what the different outputs of ϵ-PLEB(r) might be for the radii in $\mathcal{R}$.

Example 1:

Radius	$(1+\epsilon)^0$	$(1+\epsilon)^1$	$(1+\epsilon)^2$	$(1+\epsilon)^3$	$(1+\epsilon)^4$	$\cdots$	Δ
Output	No	No	No	Yes	Yes	$\cdots$	Yes
Case	1	1	2	3	3	$\cdots$	3

Example 2:

Radius	$(1+\epsilon)^0$	$(1+\epsilon)^1$	$(1+\epsilon)^2$	$(1+\epsilon)^3$	$(1+\epsilon)^4$	$\cdots$	Δ
Output	No	No	No	Yes	Yes	$\cdots$	Yes
Case	1	1	1	2	3	$\cdots$	3

Figure 25.1: Some possible outcomes of the ϵ-PLEB subroutine. Note that there can be at most one radius that results in Case 2.

Claim 25.2.2. For the radii in $\mathcal{R}$, all No outputs occur for smaller radii than all Yes outputs.

Proof. If radius r falls into Case 1, then clearly all smaller radii also fall into Case 1 and therefore produce the output No.

If radius r falls into Case 3, then clearly all larger radii also fall into Case 3 and therefore produce the output Yes.

It remains to consider Case 2. Clearly *at most one* radius in $\mathcal{R}$ can satisfy the condition of Case 2. With this radius, either Yes or No could be output, but both possibilities are consistent with the claim. □

Given any query point q, let $\hat{r}$ be the minimum radius r for which ϵ-PLEB(r) says Yes. Let $\hat{p} \in P$ be the returned point for which $q \in B(\hat{p}, (1+\epsilon)\hat{r})$. By Claim 25.2.2, $\hat{r}$ can be found by binary search.

Claim 25.2.3. $\hat{p}$ satisfies (25.2.1), and therefore is a solution to the ϵ-NN problem.

Proof. The requirements of the ϵ-PLEB$(\hat{r})$ subroutine ensure that that $\|\hat{p} - q\| \leq \hat{r}(1+\epsilon)$.

Recall that $\hat{r}$ is the *minimum* radius that said Yes. If $\hat{r} = 1$ then clearly p satisfies (25.2.1). Otherwise, ϵ-PLEB$(\hat{r}/(1+\epsilon))$ output No. It follows (from (25.2.2)) that there is no point $p' \in P$ with $q \in B(p', r/(1+\epsilon))$. In other words,

$$\frac{r}{1+\epsilon} \;<\; \min_{p' \in P} \|p' - q\| .$$

Thus $\hat{p}$ satisfies

$$\|\hat{p} - q\| \;\leq\; (1+\epsilon)\hat{r} \;<\; (1+\epsilon)^2 \cdot \min_{p' \in P} \|p' - q\| .$$

Since $(1+\epsilon)^2 \leq 1 + 3\epsilon$ (see Fact B.1.3), this proves (25.2.1). □

Question 25.2.4. How many radii must the binary search consider in order to determine r^*?

Answer.

$$\log_2|\mathcal{R}| \;=\; O\Big(\log\big(\log(\Delta)/\epsilon\big)\Big).$$

25.2.3 Solving PLEB

The main idea here is quite simple. We discretize the space, then use a hash table to identify locations belonging to a ball.

Preprocessing. In more detail, the preprocessing step for ϵ-PLEB(r) proceeds as follows. We first partition the space into cuboids (d-dimensional cubes) of side length $\epsilon r/\sqrt{d}$. Note that[1] the Euclidean diameter of a cuboid is its side length times $\sqrt{d}$, which is ϵr. Each cuboid is identified by a canonical point, say the minimal point contained in the cuboid. We then create a hash table, initially empty. For each point p_i and each cuboid C that intersects $B(p_i, r)$, we insert the $(\text{key}, \text{value})$ pair (C, p_i) into the hash table.

Queries. Now consider how to perform a query for a point q. The first step is to determine the cuboid C that contains q, by simple arithmetic. Next, we look up C in the hash table. If there are no matches, that means that no ball $B(p_i, r)$ intersects C, and therefore q is not contained in any ball of radius r (a small ball). So, by the requirements of ϵ-PLEB(r), we can say No.

Suppose that C is in the hash table. Then the hash table can return an arbitrary pair (C, p_j), which tells us that $B(p_j, r)$ intersects C. By the triangle inequality, the distance from p_j to q is at most r plus the diameter of the cuboid, which is ϵr. So $\|p_j - q\| \leq (1+\epsilon)r$, which means that q is contained in the big ball around p_j. By the requirements of ϵ-PLEB(r), we can say Yes and we can return the point p_j.

Time and Space Analysis. To analyze this algorithm, we first need to determine the number of cuboids that intersect a ball of radius r. From (B.2.3), the volume of $B(p, r)$ is $2^{O(d)} r^d / d^{d/2}$. On the other hand, the volume of a cuboid is $(\epsilon r/\sqrt{d})^d$. So the number of cuboids that intersect this ball is roughly

$$\frac{2^{O(d)} r^d / d^{d/2}}{(\epsilon r/\sqrt{d})^d} \;=\; O(1/\epsilon)^d.$$

Therefore the time and space used by the preprocessing step is roughly $O(1/\epsilon)^d$.

To perform a query, we just need to compute the cuboid containing q then look up that cuboid in the hash table. This takes $O(d)$ time if we use the perfect hashing of Section 15.4. This cannot be improved — $\Omega(d)$ time is clearly necessary, since we must examine most coordinates of the vector q.

Unfortunately the preprocessing time and space is exponential in d, which is another example of the curse of dimensionality. The next section addresses this via dimensionality reduction.

25.2.4 Applying Dimensionality Reduction

The next step is to apply the Johnson-Lindenstrauss lemma to map our points to a low-dimensional space. Let $t = O(\log(n)/\epsilon^2)$. Let R be a random $t \times d$ matrix whose entries have distribution $N(0, 1/t)$. For any fixed query point q, Theorem 24.2.1 says that the linear map R approximately preserves pairwise distances between all points in $P \cup \{q\}$ with probability $1 - 1/n$.

The analysis of ϵ-PLEB changes as follows. The preprocessing step must apply the matrix to all points in P, which takes time $O(dnt)$. The time to set up the hash table improves to $O(1/\epsilon)^t = n^{O(\log(1/\epsilon)/\epsilon^2)}$. So assuming ϵ is a constant, the preprocessing step runs in polynomial time. Each query must also apply the Johnson-Lindenstrauss matrix to the query point, which takes time $O(td) = O(d\log(n)/\epsilon^2)$.

Finally, we analyze the reduction which allowed us to solve ϵ-NN. Recall that the preprocessing step simply instantiates ϵ-PLEB(r) for all values of r in $\mathcal{R}$. As discussed in Question 25.2.1, the number of different instances is $|\mathcal{R}| = O\big(\log(\Delta)/\epsilon\big)$, so the total preprocessing time is

$$n^{O(\log(1/\epsilon)/\epsilon^2)} \;\cdot\; O\big(\log(\Delta)/\epsilon\big).$$

[1] This can be seen from Fact B.2.1, because the cuboid has ℓ_∞ diameter $\epsilon r/\sqrt{d}$.

This is polynomial time assuming $\Delta \leq 2^{n^{\text{poly}(1/\epsilon)}}$ and ϵ is a constant. Each query must perform binary search over different radii to find $\hat{r}$. Each query to ϵ-PLEB now takes time $O(t)$. So the total query time is

$$O\big(dt\big) + O(\log|\mathcal{R}|) \cdot O(t) \;=\; O\Big(t \cdot \big(d + \log(\log(\Delta)/\epsilon)\big)\Big).$$

This is roughly $O(d\log(n)/\epsilon^2)$ so long as Δ is reasonable, say $\Delta \leq 2^{2^d}$.

25.2.5 Discussion

Some of the earliest papers on approximate nearest neighbour are Kleinberg, Indyk and Motwani and Kushilevitz-Ostrovsky-Rabani. The algorithm we present is due to Indyk and Motwani. It seems quite inefficient, but of course there have been numerous improvements. The Indyk-Motwani paper itself also proposed the alternative approach of "locality sensitive hashing", which was touched upon in Section 12.3. Some modern techniques for nearest neighbor are described in the survey of Andoni and Indyk.

25.3 Fast Least-Squares Regression

Given matrix A of size $d \times m$ (with $d \geq m$) and a vector $b \in \mathbb{R}^d$, the goal is to find

$$x^* \in \operatorname*{argmin}_{x \in \mathbb{R}^m} \|Ax - b\|.$$

For some background motivation, see CPSC 340 Lecture 12 and Lecture 13.

Mathematically, this is quite a simple problem to solve: $\|Ax - b\|_2^2$ is a convex, differentiable quadratic function, so the minimizers are the points with zero gradient. After some simple vector calculus, this amounts to solving to the linear system

$$A^{\mathsf{T}}Ax = A^{\mathsf{T}}b. \tag{25.3.1}$$

References: UBC MATH 307 notes, (Trefethen and Bau, 1997, Theorem 11.1).

The focus of this section is algorithms for computationally solving (25.3.1).

Conventional Solutions. There are several standard approaches to solving (25.3.1).

- Naively, one could compute $A^{\mathsf{T}}A$ and $A^{\mathsf{T}}b$ explicitly and use Gaussian elimination to solve the problem exactly.

  Question 25.3.1. How much time does this take?

  Answer.

  $O(dm^2)$ time to compute $A^{\mathsf{T}}A$, $O(dm)$ time to compute $A^{\mathsf{T}}b$, and $O(m^3)$ time to perform Gaussian elimination. Since $d \geq m$, this is $O(dm^2)$ time.
  This can be improved to $O(dm^{1.38})$ by a galactic algorithm.

- Since $A^{\mathsf{T}}A$ is symmetric, we could use the Cholesky decomposition instead of Gaussian elimination. This approach can also solve the problem exactly in $O(dm^2)$ time.

  References: (Trefethen and Bau, 1997, Algorithm 11.1).

Theorem 25.3.2. There is a randomized algorithm to find $\hat{x}$ satisfying

$$\|A\hat{x} - b\| \;\leq\; (1 + O(\epsilon)) \cdot \|Ax^* - b\| \tag{25.3.2}$$

in time

$$O(dm \log d) + \operatorname{poly}(m \log d/\epsilon).$$

The failure probability is at most $2e^{-m}$.

The key point is that the leading term has improved from $O(dm^2)$ to only $O(dm \log d)$. This gives an improvement over conventional results if $\log d \ll m \ll d$ and ϵ is modest.

The main idea is as follows. We will use a Fast JL matrix R to reduce the number of rows of A and b to $t = O((m \log d/\epsilon)^3)$; see Section 24.4.3. After the dimension reduction, we then solve the problem by conventional methods.

Algorithm 25.2 Fast algorithm for least-squares regression. A has size $d \times m$ and b has length d.

```
1: function FastLeastSquares(matrix $A$, vector $b$, float $\epsilon$)
2:     Let $t \leftarrow O\big((m \log d/\epsilon)^3\big)$
3:     Let $R$ be a Fast JL matrix of size $t \times d$
4:     Compute the matrix $RA$ and the vector $Rb$
5:     Find $\hat{x} \in \operatorname{argmin}_{x \in \mathbb{R}^m} \|RAx - Rb\|$, by conventional methods.
6:     return $\hat{x}$
7: end function
```

Runtime analysis. The key computations are on line 4. Computing RA by ordinary matrix multiplication would take time $O(tdm)$, which is too slow. However, since R is a Fast JL matrix, we can do this much more quickly. Recall from Theorem 24.3.2 that matrix-vector multiplication with R takes time $O(d \log d + t)$. So we can construct RA by separately multiplying R by each column of A. Since there are m columns, this takes $O(dm \log d + tm)$ time. The vector Rb can be computed in the same way.

Next, on line 5, we can find $\hat{x}$ by conventional methods in $O(tm^2)$ time. Plugging in the definition of t, the total runtime is

$$O(dm \log d + tm) + O(tm^2) \;=\; O(dm \log d) + \operatorname{poly}(m \log d/\epsilon).$$

Analysis. Define

$$U \;=\; \operatorname{span}\big(\{\, Ax \,:\, x \in \mathbb{R}^m \,\} \cup \{b\}\big). \tag{25.3.3}$$

The vectors in U all have length d, but U is a subspace of dimension at most $m+1$. By (24.4.6), the matrix R approximately preserves the norm of all vectors in U, with probability at least $1 - 2e^{-m}$. By definition of $\hat{x}$, we have

$$\|RA\hat{x} - Rb\| \;\leq\; \|RAx^* - Rb\| \;=\; \|R(Ax^* - b)\| \;\leq\; (1+\epsilon) \cdot \|Ax^* - b\|,$$

since $Ax^* - b \in U$. On the other hand,

$$\|RA\hat{x} - Rb\| \;=\; \|R(A\hat{x} - b)\| \;\geq\; (1-\epsilon)\, \|A\hat{x} - b\|,$$

since $A\hat{x} - b \in U$. Putting these inequalities together, we obtain

$$\|A\hat{x} - b\| \;\leq\; \frac{1+\epsilon}{1-\epsilon} \cdot \|Ax^* - b\|.$$

Since $\frac{1}{1-\epsilon} \leq 1 + 2\epsilon$ (see Fact B.3.5), this completes the proof of Theorem 25.3.2.

Discussion. This algorithm is due to Tamas Sarlos. See also Section 10.3 of this survey.

There have been many improvements. Let nnz(A) denote the number of non-zero entries in the matrix A. It is known that the guarantee (25.3.2) can be achieved in time

$$O(\mathrm{nnz}(A)) + \mathrm{poly}(m/\epsilon).$$

See, for example, this paper or Theorem 21 of this survey.

25.4 Approximate Matrix Multiplication

Let A and B be matrices of size $n \times n$. Computing the product AB exactly takes $O(n^3)$ time by conventional methods, or $O(n^{2.38})$ by a galactic algorithm. The following algorithm is much faster, and is based on a very simple idea:

reduce the dimension of the matrices before multiplying them.

Algorithm 25.3 Algorithm to estimate AB. The matrices A and B have size $n \times n$.

```
1: function FastMatMul(matrix $A$, $B$, float $\epsilon$)
2:     Set $t = O(\log(n)/\epsilon^2)$
3:     Let $R$ be a $t \times n$ matrix with independent entries drawn from $N(0, 1/t)$
4:     return $AR^\mathsf{T}RB$
5: end function
```

Question 25.4.1. What is the runtime of this algorithm?

Answer.

It depends on the order in which the matrices are multiplied. Multiplying $R^\mathsf{T}R$ first will give runtime $O(n^3)$. However, first multiplying AR^T, then RB takes time $O(n^2t)$, and finally combining them takes time $O(n^2t) = O(n^2\log(n)/\epsilon^2)$.

One appealing aspect of this algorithm is the following claim.

Claim 25.4.2. FastMatMul(A, B, ϵ) is an unbiased estimator of AB.

The proof of this claim is Exercise 25.1.

Notation. The entry of A in the i^{th} row and j^{th} column is denoted $A_{i,j}$. To refer to rows and columns, we will use the following handy notation. Thinking of $*$ as a "wildcard" character, we will let $A_{i,*}$ denote the i^{th} row of A and let $A_{*,k}$ denote the k^{th} column of A. Using this notation, the entries of the product AB can be expressed as a dot product as follows.

$$(AB)_{i,k} = \sum_{j=1}^{n} A_{i,j}B_{j,k} = A_{i,*}B_{*,k}$$

References: (Murphy, 2022, Section 7.2.3).

Definition 25.4.3. For a matrix A, its ***Frobenius norm*** $\|A\|_F$ is defined by

$$\|A\|_F^2 = \sum_{i,j\in[n]} (A_{i,j})^2 = \sum_{i\in[n]} \|A_{i,*}\|_2^2 = \sum_{j\in[n]} \|A_{*,j}\|_2^2 . \qquad (25.4.1)$$

This is just the Euclidean norm of A when viewed as a vector of length n^2.

References: (Trefethen and Bau, 1997, page 22), (Vershynin, 2018, Section 4.1.3), (Murphy, 2022, Section 7.1.3.2), (Blum et al., 2018, Section 12.8.5), Wikipedia.

Our main theorem is that $\textsc{FastMatMul}(A, B, \epsilon)$ is a good approximation for AB, with error measured in the Frobenius norm.

Theorem 25.4.4. With probability at least $1 - 1/n$,

$$\|AB - AR^{\mathsf{T}}RB\|_F \;\leq\; \epsilon \|A\|_F \|B\|_F .$$

Proof. We apply dimensionality reduction to the rows of A and the columns of B. Define

$$\mathcal{X} \;=\; \{\, (A_{i,*})^{\mathsf{T}} \,:\, i \in [n] \,\} \cup \{\, B_{*,k} \,:\, k \in [n] \,\} .$$

Exercise 24.2 shows that, with probability $1 - 1/n$,

$$\begin{aligned} & |x^{\mathsf{T}}x' - x^{\mathsf{T}}R^{\mathsf{T}}Rx'| \;\leq\; \epsilon \|x\| \|x'\| && \forall x,\, x' \in \mathcal{X} \\ \Longrightarrow \quad & |A_{i,*}B_{*,k} - A_{i,*}R^{\mathsf{T}}RB_{*,k}| \;\leq\; \epsilon \|A_{i,*}\| \|B_{*,k}\| && \forall i,\, k \in [n]. \end{aligned}$$

Squaring and summing over i, k, we have

$$\begin{aligned} \sum_{i,k \in [n]} \big(\underbrace{A_{i,*}B_{*,k} - A_{i,*}R^{\mathsf{T}}RB_{*,k}}_{=(AB - AR^{\mathsf{T}}RB)_{i,k}} \big)^2 \;&\leq\; \epsilon^2 \sum_{i,k \in [n]} \|A_{i,*}\|^2 \|B_{*,k}\|^2 \\ &=\; \epsilon^2 \Big(\sum_{i \in [n]} \|A_{i,*}\|^2 \Big) \Big(\sum_{k \in [n]} \|B_{*,k}\|^2 \Big). \end{aligned}$$

The theorem follows by taking the square root and using the identity (25.4.1). □

25.5 Exercises

Exercise 25.1. Prove Claim 25.4.2.

Chapter 26

Online Learning

26.1 The model

We will consider an online learning model called "prediction with expert advice". This setting involves an algorithm, an adversary, and n "experts". At each time step $t = 1, 2, \ldots$, the algorithm and the experts must make a decision, and they each incur the cost of their decisions. For example, they might try to predict if some event occurs, and the cost might indicate whether their prediction was correct. These costs are controlled by the adversary. At the end of each time step, the adversary reveals information about the costs to the algorithm.

26.2 Finding a perfect expert: the halving algorithm

Let us consider a basic setting. Suppose at every time t, we are trying to predict whether a sports team will win. Each expert i announces a prediction $p_{t,i}$ of whether they will win or not. The algorithm receives these predictions then formulates its own prediction. Next, the adversary decides whether the team wins or not, and reveals this outcome.

Theorem 26.2.1. Suppose that there is some expert that is always correct. Then Algorithm 26.1 makes at most $\log_2 n$ mistakes.

Proof. Every time the algorithm makes a mistake, the majority vote over S was incorrect. This means that, in the next iteration, the size of S will have shrunk by a factor of at least 2. However, since we assume that some expert is always correct, the set S always has size at least 1. Thus, there can be at most $\log_2 n$ mistakes. $\square$

26.3 No perfect expert: the weighted majority algorithm

Now let us consider the more general case in which there is no guarantee that an expert makes perfect predictions. Let us now use the notation $c_{t,j} \in \{0, 1\}$ to denote the cost incurred by expert j at time t. That is $c_{t,j} = 0$ if their prediction was correct, otherwise $c_{t,j} = 1$. Additionally, let $C_{T,j} = \sum_{t=1}^{T} c_{t,j}$ denote the total cost incurred by expert j up to the end of iteration T.

Algorithm 26.1 The halving algorithm.

```
1: procedure HalvingAlgorithm
2:     Let $S = [n]$ (the set of experts predicting correctly so far)
3:     for $t \leftarrow 1, 2, \ldots$
4:         ▷ Receive predictions $p_t \in \{0,1\}^n$
5:         Let $m$ be the majority prediction of experts in $S$
6:         ▷ Predict $m$
7:         ▷ Receive the actual outcome, which tells us which experts were correct
8:         Remove from $S$ each expert $j$ whose prediction was incorrect
```

We now present the *weighted majority* (WM) algorithm. Instead of discarding experts that make a mistake, this algorithm will associate a "weight" with each expert: higher weight means that their predictions are more credible. The main idea is to decrease each expert's weight *multiplicatively* whenever he or she makes a mistake.

To make this precise, we need some more notation. The weight associated with expert j at time t will be denoted $y_{t,j}$. Initially all weights are $y_{1,j} = 1$, and they never increase. Every time expert j makes a mistake, his or her weight is multiplied by $e^{-\eta}$. Thus, expert j's weight at the start of iteration $T+1$ has the expression

$$y_{T+1,j} \;=\; \prod_{t=1}^{T} e^{-\eta c_{t,j}} \;=\; \exp(-\eta C_{T,j}). \tag{26.3.1}$$

Algorithm 26.2 The weighted majority algorithm.

```
1: procedure WeightedMajority
2:     Let $y_1 = [1, \ldots, 1]$
3:     for $t \leftarrow 1, 2, \ldots$
4:         ▷ Receive predictions $p_t \in \{0,1\}^n$
5:         Let $m$ be such that $\sum_{\text{expert } j \text{ predicting } m} y_{t,j} \geq \frac{1}{2} \sum_{j=1}^{n} y_{t,j}$    ▷ the weighted majority vote
6:         ▷ Predict $m$
7:         ▷ Receive the actual outcome, which determines the cost vector $c_t \in \{0,1\}^n$
8:         for $j \leftarrow 1, \ldots, n$
9:             $y_{t+1,j} = y_{t,j} e^{-\eta c_{t,j}} = \begin{cases} y_{t,j} & \text{(weight is unchanged if expert } j \text{ predicted correctly)} \\ y_{t,j} e^{-\eta} & \text{(weight is decreased if expert } j \text{ made a mistake)} \end{cases}$
```

The notation becomes simpler if we introduce a new parameter ϵ. Define

$$\epsilon = 1 - e^{-\eta} \qquad \text{or equivalently} \qquad \eta = \ln \frac{1}{1-\epsilon}. \tag{26.3.2}$$

The parameters η and ϵ are closely related, since one can show

$$\epsilon \approx \eta - \eta^2/2 \qquad \text{and} \qquad \eta \approx \epsilon + \epsilon^2/2.$$

We will furthermore assume that $\epsilon \leq 1/2$. Then, using Fact B.3.6, we obtain that

$$\eta \leq \epsilon + \epsilon^2. \tag{26.3.3}$$

Theorem 26.3.1. Assume that $\epsilon \leq 1/2$. Consider any time step T. Let A_T be the total cost (i.e., number of mistakes) of the algorithm after T iterations. Then, for every expert $j \in [n]$,

$$A_T \;\leq\; 2(1+\epsilon)C_{T,j} + \frac{2\ln n}{\epsilon}.$$

In particular, this holds even for the "best expert", i.e., $j^* \in \operatorname{argmin}_j C_{T,j}$.

References: (Hazan, 2015, Lemma 1.3).

Remark 26.3.2. There are variants of this algorithm that scale the weights by $(1-\eta c_{i,j})$ or $(1-\eta)^{c_{i,j}}$ instead of $\exp(-\eta c_{i,j})$. The proof can be easily modified to use those scalings instead.

Idea. The proof has two main ideas.

(a) The cost incurred by the algorithm in iteration t can be related to the total change in the weights during iteration t.

(b) The total weight in iteration $T+1$ provides a useful upper bound on the total cost that any expert has incurred up to iteration T.

Proof.

Step (a): We must analyze the cost of the algorithm in iteration t. The main fact at our disposal is: whenever the algorithm makes a mistake, the weighted majority vote was incorrect. In mathematical notation, $\sum_{\text{mistaken } j} y_{t,j} \geq \frac{1}{2}\sum_j y_{t,j}$. Since the mistaken experts' weights are decreased by a factor $e^{-\eta}$, the main consequence is as follows.

$$\text{If algorithm makes a mistake:} \quad \sum_j y_{t+1,j} = \sum_{\text{mistaken } j} y_{t,j}e^{-\eta} + \sum_{\text{correct } j} y_{t,j}$$
$$= \sum_j y_{t,j} \;-\; (1-e^{-\eta}) \underbrace{\sum_{\text{mistaken } j} y_{t,j}}_{\geq \sum_j y_{t,j}/2}$$
$$\leq \left(1 - \frac{1-e^{-\eta}}{2}\right)\sum_j y_{t,j}$$
$$= \left(1-\frac{\epsilon}{2}\right)\cdot\sum_j y_{t,j} \quad \text{(by definition of } \epsilon\text{)}.$$

Using the bound $1+x \leq e^x$, we obtain the bound

$$\frac{\|y_{t+1}\|_1}{\|y_t\|_1} \;\leq\; \begin{cases} 1 & \text{(if algorithm is correct)} \\ e^{-\epsilon/2} & \text{(if algorithm makes a mistake).} \end{cases}$$

Taking the product for $t = 1,\ldots,T$, we obtain

$$\frac{\|y_{T+1}\|_1}{\|y_1\|_1} = \prod_{t=1}^{T} \frac{\|y_{t+1}\|_1}{\|y_t\|_1} \leq \prod_{t\,:\,\text{algorithm makes a mistake}} e^{-\epsilon/2} = \exp(-\epsilon A_T/2),$$

since A_T is the number of mistakes made by the algorithm. Thus, since $\|y_1\|_1 = n$, we have

$$\|y_{T+1}\|_1 \;\leq\; n\cdot\exp(-\epsilon A_T/2). \tag{26.3.4}$$

Step (b): The next idea is that the sum of the weights gives a reasonable bound on any individual weight. Specifically, by (26.3.1), we have

$$\exp(-\eta C_{T,j}) \;=\; y_{T+1,j} \;\leq\; \|y_{T+1}\|_1 \,. \tag{26.3.5}$$

Combining (a) and (b): To relate the cost of expert j and the cost of the algorithm, we combine (26.3.4) and (26.3.5) to obtain

$$\exp(-\eta C_{T,j}) \;\leq\; \|y_{T+1}\|_1 \;\leq\; \exp(-\epsilon A_T/2)\cdot n.$$

Taking the log, multiplying by $\frac{2}{\epsilon}$ and rearranging, we obtain

$$A_T \;\leq\; \frac{2\eta}{\epsilon}C_{T,j} + \frac{2\ln n}{\epsilon} \;=\; \frac{2}{\epsilon}\ln\Big(\frac{1}{1-\epsilon}\Big)C_{T,j} + \frac{2\ln n}{\epsilon}.$$

Lastly, using (26.3.3), the coefficient of $C_{T,j}$ is bounded by $\frac{2}{\epsilon}(\epsilon+\epsilon^2) = 2(1+\epsilon)$, which completes the proof. □

Remark 26.3.3. In this analysis, the current total weight of the experts, namely $\sum_j y_{t,j} = \|y_t\|_1$, plays an important role. One may view this quantity as a "potential function" that captures the state of the algorithm. Step (a) relates this potential to the cost of the algorithm. Step (b) relates this potential to the cost of the best expert.

26.3.1 Lower bound

Let us consider a simple scenario with only two experts, where the first expert always predicts that the team will win, and the second expert always predicts that they will lose. Suppose that the algorithm can be any deterministic algorithm. If the adversary knows which algorithm is being used, then the adversary knows what the algorithm will predict in every iteration. Thus, in every iteration, the adversary can choose the sports team's outcome so that the algorithm makes a mistake. However, only one of the two experts mistakes a mistake. In mathematical symbols, we have

$$A_T = T \quad\text{and}\quad C_{T,1} + C_{T,2} = T. \tag{26.3.6}$$

From this we obtain the following claim.

Claim 26.3.4. Consider a prediction scenario with two experts and with *any* deterministic algorithm. Then, there is an adversary that ensures that

$$A_T \geq 2C_{T,j^*},$$

where $j^* \in \operatorname{argmin}_j C_{T,j}$ is the "best expert".

Proof. From (26.3.6) we have $C_{T,j^*} = \min_{j\in[2]} C_{T,j} \leq T/2 = A_T/2$. □

We conclude that no deterministic algorithm can remove the constant factor of 2 from Theorem 26.3.1.

26.4 The randomized weighted majority algorithm

In this section we introduce an extremely interesting algorithm: the *randomized weighted majority* (RWM) algorithm.

The first interesting result we will show is that it can remove the undesirable factor of 2 from Theorem 26.3.1 — something that no deterministic algorithm can do! We will show that RWM's *expected* cost can be bounded by roughly $(1+\epsilon)C_{T,j^*}$.

There is another remarkable property of this algorithm. Whereas WM needs to receive the experts' predictions in order to decide its own prediction (see Algorithm 26.2), RWM can decide what do *before* even seeing the experts' predictions. Instead RWM will, in a specific random way, pick an expert to "follow". Then, at the end of the iteration, it needs to know what cost every expert incurred.

Algorithm 26.3 The randomized weighted majority algorithm.

```
1: procedure RandomizedWeightedMajority($\eta$)
2:     Let $y_1 \leftarrow [1,\ldots,1]$
3:     for $t \leftarrow 1,2,\ldots$
4:         Let $x_t \leftarrow y_t / \|y_t\|_1$ (normalize the weights).
5:         Follow expert $j$ with probability $x_{t,j}$.
6:         ▷ The expected cost incurred by the algorithm is $\sum_{j=1}^n c_{t,j}x_{t,j} = \langle c_t, x_t\rangle$
7:         ▷ Receive cost vector $c_t$ from the adversary
8:         for $j \leftarrow 1,\ldots,n$
9:             $y_{t+1,j} \leftarrow y_{t,j}\exp(-\eta c_{t,j})$        ▷ decrease weight according to expert $j$'s cost
```

The quantities $\epsilon = 1 - e^{-\eta}$ and $C_{T,j} = \sum_{t=1}^T c_{t,j}$ are defined as in the previous section. Now

$$A_T \;=\; \sum_{t=1}^T \langle c_t, x_t\rangle$$

will denote the total *expected* cost of the algorithm,

Theorem 26.4.1. *Assume that $c_t \in [0,1]^n$ for all t. Assume that $\epsilon \leq 1/2$. Consider any time step T. Then, for every expert $j \in [n]$,*

$$A_T \;\leq\; (1+\epsilon)C_{T,j} + \frac{\ln n}{\epsilon}.$$

In particular, this holds even for the "best expert", i.e., $j^* \in \operatorname{argmin}_j C_{T,j}$.

References: (Hazan, 2015, Lemma 1.4), (Shalev-Shwartz and Ben-David, 2014, Theorem 21.11).

Idea. As in the proof of WM, it is useful to think of $\|y_t\|_1$ as a potential function. Step (b) is unchanged: the total cost of expert j is bounded using the potential. Step (a) is a bit different: we will relate the change in potential $\|y_{t+1}\|_1 / \|y_t\|_1$ to the *expected* cost incurred by the algorithm at time step t. In doing so, we will avoid the unwanted factor 2 that arises with WM.

Proof.

Step (a):

$$\frac{\|y_{t+1}\|_1}{\|y_t\|_1} \;=\; \sum_{j=1}^n \frac{y_{t,j}\exp(-\eta c_{t,j})}{\|y_t\|_1} \;\leq\; \sum_{j=1}^n x_{t,j}(1-\epsilon c_{t,j}) \;=\; 1 - \epsilon\sum_{j=1}^n x_{t,j}c_{t,j} \;\leq\; \exp(-\epsilon\langle c_t, x_t\rangle).$$

The first inequality uses Fact B.3.8 with $\alpha = e^{-\eta}$ and the definition $\epsilon = 1 - e^{-\eta}$. The second inequality is Fact A.2.5. Thus, taking the product for $t = 1, \ldots, T$, we see that the total weight at step $T+1$ is related to the total expected cost incurred by the algorithm:

$$\frac{\|y_{T+1}\|_1}{\|y_1\|_1} \;=\; \prod_{t=1}^{T} \frac{\|y_{t+1}\|_1}{\|y_t\|_1} \;\leq\; \prod_{t=1}^{T} \exp(-\epsilon\langle c_t, x_t\rangle) \;=\; \exp\Big(-\epsilon \underbrace{\sum_{t=1}^{T}\langle c_t, x_t\rangle}_{=A_T}\Big).$$

Thus, recalling that $\|y_1\|_1 = n$, we obtain

$$\|y_{T+1}\|_1 \;\leq\; n\exp(-\epsilon A_T). \tag{26.4.1}$$

Step (b): As in (26.3.5), we have

$$\exp(-\eta C_{T,j}) \;=\; y_{T+1,j^*} \;\leq\; \|y_{T+1}\|_1\,. \tag{26.4.2}$$

Combining (a) and (b): To relate the cost of expert j and the expected cost of the algorithm, we combine (26.4.1) and (26.4.2) to obtain

$$\exp(-\eta C_{T,j}) \;\leq\; \|y_{T+1}\|_1 \;\leq\; n \cdot \exp(-\epsilon A_T).$$

Taking the log and rearranging, we obtain

$$A_T \;\leq\; \frac{\eta}{\epsilon} C_{T,j} \;+\; \frac{\ln n}{\epsilon}.$$

Lastly, using (26.3.3), the coefficient of $C_{T,j}$ is bounded by $\frac{1}{\epsilon}(\epsilon + \epsilon^2) \;=\; (1+\epsilon)$, which completes the proof. □

26.4.1 Extensions

Regret bounds

A fundamental concept in online learning is the notion of ***regret***. In our setting, regret is defined to be the difference between the algorithm's cost and the cost of the best expert:

$$\text{Regret}(T) \;=\; A_T - \min_{j\in[n]} C_{T,j}.$$

If one wishes to achieve low regret at time step T, one may optimize ϵ as a function of T and obtain the following result.

Corollary 26.4.2. Set $\epsilon = \sqrt{\ln(n)/T}$. Then $\text{Regret}(T) \;\leq\; 2\sqrt{T\ln n}$.

Proof. For any $j \in [n]$, we have

$$\begin{aligned} A_T - C_{T,j} \;&\leq\; \epsilon C_{T,j} + \frac{\ln n}{\epsilon} \qquad &&\text{(by Theorem 26.4.1)}\\ &\leq\; \epsilon T + \frac{\ln n}{\epsilon} &&\text{(since } c_t \in [0,1]^n\ \forall t)\\ &=\; 2\sqrt{T\ln n}, \end{aligned}$$

by choice of ϵ. In particular, this holds for $j^* \in \operatorname{argmin}_j C_{T,j}$, which implies the regret bound. □

Remark 26.4.3. The constant factor 2 can be improved to $2^{-1/2}$, which is optimal. See (Cesa-Bianchi and Lugosi, 2006, Section 2.2).

Non-negative costs, quadratic bound

In this section we assume only that the cost vectors c_t are non-negative.

Theorem 26.4.4. For every $j \in [n]$, we have

$$A_T \;\leq\; \sum_{t=1}^{T} c_{t,j} + \frac{\eta}{2} \sum_{t=1}^{T} \sum_{k \in [n]} x_{t,k} c_{t,k}^2 + \frac{\ln n}{\eta}.$$

Taking the minimum over j, we obtain the regret bound

$$\operatorname{Regret}(T) \;\leq\; \frac{\eta}{2} \sum_{t=1}^{T} \sum_{k \in [n]} x_{t,k} c_{t,k}^2 + \frac{\ln n}{\eta}.$$

Proof. We simply change step (a) of the proof of Theorem 26.4.1 as follows:

$$\begin{aligned}
\frac{\|y_{t+1}\|_1}{\|y_t\|_1} &= \sum_{k=1}^{n} \frac{y_{t,k} \exp(-\eta c_{t,k})}{\|y_t\|_1} \\
&= \sum_{k=1}^{n} x_{t,k} \exp(-\eta c_{t,k}) \\
&\leq \sum_{k=1}^{n} x_{t,k} \Big(1 - \eta c_{t,k} + \frac{\eta^2}{2} c_{t,k}^2 \Big) \qquad \text{(by Fact B.3.7, and since } c_{t,k} \geq 0\text{)} \\
&\leq 1 + \Big(-\eta \langle c_t, x_t \rangle + \frac{\eta^2}{2} \sum_{k=1}^{n} x_{t,k} c_{t,k}^2 \Big) \\
&\leq \exp\Big(-\eta \langle c_t, x_t \rangle + \frac{\eta^2}{2} \sum_{k=1}^{n} x_{t,k} c_{t,k}^2 \Big) \qquad \text{(by Fact A.2.5).}
\end{aligned}$$

Now, as before, we combine steps (a) and (b) to obtain:

$$\exp(-\eta C_{T,j}) \;\leq\; \|y_{T+1}\|_1 \;\leq\; n \cdot \exp\Big(-\eta \underbrace{\sum_{t=1}^{T} \langle c_t, x_t \rangle}_{=A_T} + \frac{\eta^2}{2} \sum_{t=1}^{T} \sum_{k=1}^{n} x_{t,k} c_{t,k}^2 \Big).$$

Taking the log, dividing by η and rearranging yields the claimed bound. $\square$

Arbitrary costs

Next suppose we also allow negative costs. It is no longer possible to obtain multiplicative error, but we can obtain a bound with an additive error term of $O(\epsilon T)$.

Theorem 26.4.5. Assume that the costs satisfy $c_t \in [-\rho, \rho]^n$ for all t. Let $\epsilon = 1 - e^{-\eta}$ and assume that $\epsilon \leq 1/2$. Consider any time step t. For any $j \in [n]$, the expected cost of the algorithm satisfies

$$A_T \;\leq\; C_{T,j} + 2\rho\Big(\frac{\ln n}{\epsilon} + \epsilon T \Big).$$

Proof. The idea is to transform the costs into the interval $[0,1]$, then to apply Theorem 26.4.1. Let $\vec{1}$ denote the all-ones vector. We define

$$\hat{c}_t = \frac{1}{2\rho}c_t + \frac{1}{2}\vec{1}, \qquad \text{or equivalently} \qquad 2\rho\hat{c}_t - \rho\vec{1} = c_t. \tag{26.4.3}$$

We then apply Theorem 26.4.1 with the cost vectors $\hat{c}_1, \ldots, \hat{c}_T$. This leads to the bound

$$\begin{aligned}
\sum_{t=1}^{T}\langle c_t, x_t\rangle &= 2\rho\sum_{t=1}^{T}\langle \hat{c}_t, x_t\rangle - \rho T && \text{(by (26.4.3), and using } \langle \vec{1}, x_t\rangle = 1)\\
&\leq 2\rho(1+\epsilon)\sum_{t=1}^{T}\hat{c}_{t,j} + \frac{2\rho\ln n}{\epsilon} - \rho T && \text{(by Theorem 26.4.1)}\\
&= \Big(2\rho\sum_{t=1}^{T}\hat{c}_{t,j} - \rho T\Big) + 2\rho\epsilon\sum_{t=1}^{T}\hat{c}_{t,j} + \frac{2\rho\ln n}{\epsilon}\\
&\leq \sum_{t=1}^{T}\underbrace{(2\rho\hat{c}_{t,j} - \rho)}_{=c_{t,j}} + 2\rho\epsilon T + \frac{2\rho\ln n}{\epsilon} && \text{(since } \hat{c}_{t,j} \leq 1).
\end{aligned}$$

Thus we have shown that

$$\underbrace{\sum_{t=1}^{T}\langle c_t, x_t\rangle}_{=A_T} \leq \underbrace{\sum_{t=1}^{T}c_{t,j}}_{=C_{T,j}} + 2\rho\epsilon T + \frac{2\rho\ln n}{\epsilon},$$

which is the claimed bound. □

Remark 26.4.6. The ϵT term can be improved to $\epsilon\sum_{t=1}^{T}|c_{t,j^*}|$. See (Arora et al., 2012, Theorem 3).

Corollary 26.4.7. Let $\delta \in (0,1]$ be arbitrary. Let $\rho \geq 1$ be such that the costs satisfy $c_t \in [-\rho,\rho]^n$ for all $t \in [T]$. Let $T \geq \frac{16\rho^2\ln n}{\delta^2}$, $\epsilon = \sqrt{\ln(n)/T}$ and $\eta = \ln\frac{1}{1-\epsilon}$. Then, for any $j \in [n]$,

$$\frac{A_T}{T} \leq \frac{C_{T,j}}{T} + \delta.$$

Proof. From Theorem 26.4.5 we obtain

$$\sum_{t=1}^{T}\frac{\langle c_t, x_t\rangle}{T} \leq \sum_{t=1}^{T}\frac{c_{t,j}}{T} + 2\rho\Big(\frac{\ln n}{\epsilon T} + \epsilon\Big) = \sum_{t=1}^{T}\frac{c_{t,j}}{T} + 4\rho\frac{\sqrt{\ln n}}{\sqrt{T}} \leq \sum_{t=1}^{T}\frac{c_{t,j}}{T} + \delta.$$

This is the claimed bound. □

Exercises

Exercise 26.1. Is Regret(t) a monotonically non-decreasing function?

Exercise 26.2. Suppose we know in advance a value C^* such that some expert j has cost $C_{T,j} \leq C^*$. (Of course, we do not know which j.) Describe an algorithm that can achieve regret at most $2\sqrt{C^*\ln n}$ and explain how it achieves this bound.

Exercise 26.3. Intuitively, if the algorithm incurs a large expected cost then experts, on aggregate, also incur a large cost. Therefore, there should be few experts that have incurred very little cost.

Recall that A_T denotes the total cost incurred by the algorithm up to time T. Prove that, at time T, the number of experts with total cost at most C is at most

$$n \exp\big((\epsilon + \epsilon^2)C - \epsilon A_T\big).$$

Chapter 27

Applications of RWM

In this chapter we cover several interesting applications of the randomized weighted majority algorithm.

27.1 Linear programming

Consider the following linear program with variables $v \in \mathbb{R}^n$, constraint matrix A of size $m \times n$, and right-hand side vector $b \in \mathbb{R}^m$.

$$\begin{array}{lll} \min & f^{\mathsf{T}} v & \\ \text{s.t.} & Av & \geq b \\ & v \in \mathcal{C} & \end{array}$$

The objective function is determined by the vector $f \in \mathbb{R}^n$. The variables are constrained to lie in a convex set $\mathcal{C}$, which captures the "easy" constraints of the problem. What qualifies as an easy constraint depends on the scenario, and ultimately it will relate to the "oracle" that we design below. As a typical example to keep in mind, we can imagine that

$$\mathcal{C} \;=\; \{\, v \in \mathbb{R}^n \,:\, l_j \leq v_j \leq u_j \; \forall j \in [n] \,\},$$

where the vector $l \in \mathbb{R}^n$ gives lower bounds on the variables, and the vector $u \in \mathbb{R}^n$ gives upper bounds on the variables.

To keep things simple, we will ignore the objective function and consider only the following *feasibility problem*. For convenience, let us A_i denote the i^{th} row of A. Let

$$\mathcal{F} \;=\; \{\, v \in \mathbb{R}^n \,:\, A_i v \geq b_i \; \forall i \in [m],\; v \in \mathcal{C} \,\}.$$

The goal of the feasibility problem is to find $v \in \mathcal{F}$, or determine that $\mathcal{F} = \emptyset$.

Remark 27.1.1. The problem of optimizing the original linear program can be reduced to the feasibility problem. For example, one approach is to add a new constraint $f^{\mathsf{T}} v \leq \alpha$ (equivalently, $-f^{\mathsf{T}} v \geq -\alpha$), then to use binary search to find a near-optimal value of α. Other reductions are known that avoid this binary search.

We will solve the feasibility problem only approximately. For $\delta > 0$, define the δ-relaxation of the feasible region to be

$$\mathcal{F}_\delta \;=\; \{\, v \in \mathbb{R}^n \,:\, A_i v \geq b_i - \delta \; \forall i \in [m],\; v \in \mathcal{C} \,\}.$$

Our algorithm will either output a point $\hat{v} \in \mathcal{F}_\delta$, or determine that $\mathcal{F} = \emptyset$.

The Oracle. We will assume that we have an "oracle" (or subroutine) that behaves as follows. Its input is a vector $w \in \mathbb{R}^n$ and a scalar $d \in \mathbb{R}$. The output has two possibilities.

- **"Yes":** the output is a point $v \in \mathcal{C}$ satisfying $w^\mathsf{T} v \geq d$.
- **"No":** the oracle asserts that no such point exists.

Furthermore, we assume that there is a parameter $\rho > 0$ such that, whenever the oracle answers "Yes", the point v satisfies

$$A_i v - b_i \in [-\rho, \rho] \qquad \forall i \in [m]. \tag{27.1.1}$$

This parameter ρ is called the ***width***.

To simplify matters for the oracle, its inputs w and d will not be arbitrary. The constraint "$w^\mathsf{T} v \geq d$" that the oracle attempts to satisfy will always be a convex combination of the original constraints "$A_i v \geq b_i$". More explicitly, define the probability simplex in $\mathbb{R}^m$ to be

$$\Delta_m \;=\; \{\, x \in \mathbb{R}^m \,:\, x_i \geq 0 \;\forall i \in [m] \text{ and } \textstyle\sum_{i=1}^m x_i = 1 \,\}.$$

There will always be a distribution $x \in \Delta_m$ such that

$$\text{(constraint that oracle attempts to satisfy)} \;=\; \sum_{i=1}^m x_i \cdot (i^{\text{th}} \text{ constraint of } \mathcal{F}).$$

That is, we will always have $w^\mathsf{T} = x^\mathsf{T} A$ and $d = x^\mathsf{T} b$ for some $x \in \Delta_m$.

Overview. Our LP solver algorithm combines the oracle with RWM in an interesting way. The key idea is to think of each of the m constraints as an "expert", then to use RWM to maintain a weighting over the constraints. The current weighted average of the constraints is then provided as a *single* constraint for the oracle to satisfy.

Of course, when the oracle returns "Yes", that does not necessarily mean that the constraints of $\mathcal{F}$ are satisfied. That is, in general

$$w^\mathsf{T} v \geq d \qquad \not\Rightarrow \qquad A_i^\mathsf{T} v \geq b_i.$$

(One exception is the case that $x_i = 1$: when all the weighting is on the i^{th} constraint.) Nevertheless, a larger weighting x_i on the i^{th} constraint means that we want the oracle to place more emphasis on satisfying the i^{th} constraint. Together with the regret bound of RWM, that will ultimately allow us to approximately satisfy all of the constraints of $\mathcal{F}$.

Algorithm. The pseudocode for the algorithm as follows.

Algorithm 27.1 The LP feasibility solver.

```
 1: procedure LPSolve($\delta$)
 2:     Let $T \leftarrow \frac{16\rho^2 \ln n}{\delta^2}$
 3:     Initialize RWM with parameter $\epsilon \leftarrow \sqrt{\ln(n)/T}$
 4:     for $t \leftarrow 1, \ldots, T$
 5:         Get the current distribution $x_t \in \mathbb{R}^m$ from RWM
 6:         Let $w_t \leftarrow x_t^\mathsf{T} A$ and $d_t \leftarrow x_t^\mathsf{T} b$
 7:         ▷ The current weighted average of the constraints is "$w_t^\mathsf{T} v \geq d_t$"
 8:         Invoke the oracle with inputs $w_t$, $d_t$
 9:         if oracle returns "No" then
10:             return "$\mathcal{F}$ is empty"
11:         else
12:             Receive $v_t$ from the oracle                  ▷ We are guaranteed that $w_t^\mathsf{T} v_t \geq d_t$
13:             Let $c_t \leftarrow A v_t - b$                 ▷ The cost vector $c_t$ is in $[-\rho, \rho]^m$
14:             Provide $c_t$ as the cost vector to RWM
15:     return $\frac{1}{T} \sum_{t=1}^T v_t$
```

Analysis.

Theorem 27.1.2. Algorithm 27.1 either outputs a point $\hat{v} \in \mathcal{F}_\delta$ or determines that $\mathcal{F} = \emptyset$.

References: (Arora et al., 2012, Section 3.2), (Plotkin et al., 1995), (Grigoriadis and Khachiyan, 1995).

The theorem is a consequence of the following two lemmas.

Lemma 27.1.3. If the oracle ever returns "No" then $\mathcal{F} = \emptyset$.

Proof. For the purpose of contradiction, assume that there is a point $v \in \mathcal{F}$. Then we have $A_i v - b_i \geq 0$ for all i. Now consider any time t at which the oracle returns "No". Multiplying the i^{th} inequality by the non-negative weight $x_{t,i}$ then adding up, we obtain

$$0 \;\leq\; \sum_{i=1}^m x_{t,i}(A_i v - b_i) \;=\; \underbrace{\Big(\sum_{i=1}^m x_{t,i} A_i \Big)}_{= x_t^\mathsf{T} A} v - \underbrace{\Big(\sum_{i=1}^m x_{t,i} b_i \Big)}_{= x_t^\mathsf{T} b} \;=\; w_t^\mathsf{T} v - d.$$

Furthermore, $v \in \mathcal{C}$ since $\mathcal{F} \subseteq \mathcal{C}$. However the oracle's output "No" asserts that no such point exists. This is a contradiction, so no point $v \in \mathcal{F}$ can exist. □

Lemma 27.1.4. Suppose that the oracle always returns "Yes". Let $\hat{v} = \frac{1}{T} \sum_{t=1}^T v_t$ be the algorithm's output. Then $\hat{v} \in \mathcal{F}_\delta$.

Proof. In each iteration we know that $w_t^\mathsf{T} v_t - d_t \geq 0$. This implies that the RWM algorithm's average cost is non-negative:

$$\frac{A_T}{T} \;=\; \frac{1}{T} \sum_{t=1}^T x_t^\mathsf{T} c_t \;=\; \frac{1}{T} \sum_{t=1}^T x_t^\mathsf{T} (A v_t - b) \;=\; \frac{1}{T} \sum_{t=1}^T (w_t^\mathsf{T} v_t - d_t) \;\geq\; 0, \tag{27.1.2}$$

since $w_t^\mathsf{T} = x_t^\mathsf{T} A$ and $d_t = x_t^\mathsf{T} b$. On the other hand, for any i, the average cost of the i^{th} expert is

$$\frac{C_{T,i}}{T} \;=\; \frac{1}{T} \sum_{t=1}^T c_{t,i} \;=\; \frac{1}{T} \sum_{t=1}^T (A_i v_t - b_i) \;=\; A_i \hat{v} - b_i. \tag{27.1.3}$$

By our assumption on the oracle, we have that $c_{t,i} = A_i v_t - b_i \in [-\rho, \rho]$ for all t and i. Thus, using Corollary 26.4.7, which is our regret bound for RWM with costs in $[-\rho, \rho]$, we obtain that

$$\frac{A_T}{T} \leq \frac{C_{T,i}}{T} + \delta.$$

From (27.1.2) we have $A_T/T \geq 0$, so rearranging and using (27.1.3) we get

$$-\delta \leq \frac{C_{T,i}}{T} = A_i \hat{v} - b_i.$$

This argument holds for all $i \in [m]$. Furthermore, the oracle guarantees that $v_t \in \mathcal{C}$ for all t so, by convexity of $\mathcal{C}$, we also have $\hat{v} = \frac{1}{T}\sum_{t=1}^{T} v_t \in \mathcal{C}$. Thus, we conclude that $\hat{v} \in \mathcal{F}_\delta$. $\square$

Remark 27.1.5. This approach can also handle a feasibility problem of the form

$$\mathcal{F} = \{\, v \in \mathbb{R}^n \,:\, Av \leq b,\ v \in \mathcal{C} \,\},$$

if we have an oracle to find v satisfying $w^\mathsf{T} v \leq d$. This problem reduces to Theorem 27.1.2 simply by substituting $A \leftarrow -A$ and $b \leftarrow -b$. Of course, in this case we would have

$$\mathcal{F}_\delta = \{\, v \in \mathbb{R}^n \,:\, A_i v \leq b_i + \delta\ \forall i \in [m],\ v \in \mathcal{C} \,\},$$

Bounded width. Above we have assumed that the oracle has bounded width, i.e., that there exists a finite parameter ρ such that $A_i v - b_i \in [-\rho, \rho]^m$ for all $i \in [m]$. In fact, this assumption is without loss of generality. To see why, consider the following theorem.

Theorem 27.1.6. Define the matrix $A' = \begin{pmatrix} A & I \end{pmatrix}$ and the function

$$f(A, b) = \max_B \frac{\|b\|}{\sigma_{\min}(A'_B)},$$

where A'_B denotes the subset of A' with column set B, $\sigma_{\min}$ denotes the minimum singular value, and the maximum is over subsets B such that A'_B is square and non-singular. If $\mathcal{F} \neq \emptyset$, then there exists $v \in \mathcal{F}$ satisfying

$$\|v\|_2 \leq f(A, b) \qquad \forall i \in [m].$$

Corollary 27.1.7 (Finite bound on the width)**.** Define

$$\rho = \max_i \|A_i\|_2 \cdot f(A, b) + \max_i |b_i|.$$

Then, for the particular vector v whose existence is asserted by Theorem 27.1.6, we have

$$|A_i v - b_i| \leq \rho \qquad \forall i \in [m].$$

Thus, there exists an oracle with width parameter ρ.

27.2 LP for Congestion Minimization

Recall the congestion minimization problem from Section 22.2. We have a graph $G = (V, E)$ and k source-sink pairs $s_i, t_i \in V$. We want to find an s_i-t_i path for each i such that each edge is used in at most C paths. We use $\mathcal{P}^i$ to denote the set of all s_i-t_i paths in G. We previously discussed the following linear programming formulation of this problem.

$$\begin{array}{lll} \min & C & \\ \text{s.t.} & \sum_{P \in \mathcal{P}_i} v_P^i = 1 & \forall i \in [k] \\ & \sum_i \sum_{\substack{P \in \mathcal{P}^i \\ e \in P}} v_P^i \leq C & \forall e \in E \\ & C \geq 1 & \\ & v_P^i \geq 0 & \forall i \in [k],\ P \in \mathcal{P}^i \end{array}$$

Here we have renamed the LP variables from x_P^i to v_P^i to avoid a naming conflict with the x_t vectors computed by RWM.

In this section, we will show how to solve this linear program, using the techniques of the previous section. The first step is to formulate it using as a feasibility problem using the sets $\mathcal{C}$ and $\mathcal{F}$ discussed earlier. First, the "easy" constraints are defined as follows.

$$\begin{aligned} \mathcal{C}^i &= \left\{ v^i \in \mathbb{R}^{\mathcal{P}^i} \,:\, \sum_{P \in \mathcal{P}^i} v_P^i = 1 \text{ and } v_P^i \geq 0 \forall P \in \mathcal{P}^i \right\} \\ \mathcal{C} &= \prod_{i \in [k]} \mathcal{C}^i = \left\{ v \in \prod_{i \in [k]} \mathbb{R}^{\mathcal{P}^i} \,:\, \begin{array}{l} \sum_{P \in \mathcal{P}^i} v_P^i = 1 \ \forall i \in [k] \\ v_P^i \geq 0 \ \ \forall i \in [k], \forall P \in \mathcal{P}^i \end{array} \right\} \end{aligned}$$

At this point, it is useful to keep in mind that $\mathcal{C}^i$ is a probability simplex in $\mathbb{R}^{\mathcal{P}^i}$, and that $\mathcal{C}$ is a Cartesian product of the sets $\mathcal{C}^i$.

Next, for a fixed constant $C \in \mathbb{R}$, define the feasible region

$$\mathcal{F}^C = \left\{ v \in \mathcal{C} \,:\, \sum_{i \in [k]} \sum_{\substack{P \in \mathcal{P}^i \\ e \in P}} v_P^i \leq C \ \forall e \in E \right\}$$

The linear program in Section 22.2 is equivalent to finding the minimum value C such that $\mathcal{F}^C$ is non-empty.

Our approach is to design an algorithm for the feasibility problem $\mathcal{F}^C$. Then we will use binary search to find the smallest value of $C \in [1, k]$ such that $\mathcal{F}^C$ is non-empty. To decide feasibility of $\mathcal{F}^C$ we will use Algorithm 27.1, which requires us to design an oracle.

The Oracle. The oracle is given a convex combination of the constraints, where the constraint for edge e has weight $x_{t,e}$. This convex combination can be expressed as

$$\sum_{e \in E} x_{t,e} \underbrace{\Big(\sum_{i \in [k]} \sum_{\substack{P \in \mathcal{P}^i \\ e \in P}} v_P^i \Big)}_{\substack{\text{left-hand side} \\ \text{of } e^{\text{th}} \text{ constraint}}} \leq \sum_{e \in E} x_{t,e} \underbrace{C}_{\substack{\text{right-hand side} \\ \text{of } e^{\text{th}} \text{ constraint}}} \tag{27.2.1}$$

We now restate this in an equivalent form. First, the right-hand side of (27.2.1) is clearly C since x_t is a distribution. Next, on the left-hand side, we can change the order of summation to $\sum_i \sum_P \sum_e$. And finally, the existence of a vector v satisfying (27.2.1) can be stated as minimizing the left-hand side over v, then verifying whether that minimum value is at most C. Thus, the condition that the oracle must evaluate is

$$\min_{v \in \mathcal{C}} \sum_{i \in [k]} \sum_{P \in \mathcal{P}^i} v_P^i \underbrace{\Big(\sum_{e \in P} x_{t,e} \Big)}_{= \text{length}_{x_t}(P)} \overset{?}{\leq} C. \tag{27.2.2}$$

Here we introduce the notation $\text{length}_{x_t}(P) := \sum_{e \in P} x_{t,e}$ to denote the total length of the path P when using $x_{t,e}$ as the length of edge e. Lastly, we observe that the commodities do not interact at all, neither in the function $\sum_i \sum_P v_P^i \,\text{length}_{x_t}(P)$ (since it is separable), nor in the set $\mathcal{C}$ (since it has a product structure $\mathcal{C} = \prod_{i=1}^k \mathcal{C}^i$). Thus, the separation principle (Claim B.1.4) implies that (27.2.2) is equivalent to

$$\sum_{i \in [k]} \min_{v^i \in \mathcal{C}^i} \sum_{P \in \mathcal{P}^i} v_P^i \,\text{length}_{x_t}(P) \overset{?}{\leq} C. \tag{27.2.3}$$

Recall that $\mathcal{C}^i$ imposes the constraint that v^i is a distribution over paths in $\mathcal{P}^i$. Thus, the inner minimization is minimizing a convex combination of path lengths, which must be achieved achieved by a single path P. In symbols,

$$\min_{v^i \in \mathcal{C}^i} \sum_{P \in \mathcal{P}^i} v_P^i \,\text{length}_{x_t}(P) = \min_{P \in \mathcal{P}^i} \text{length}_{x_t}(P).$$

To conclude, the goal of the oracle is to determine whether

$$\sum_{i \in [k]} \min_{P \in \mathcal{P}^i} \text{length}_{x_t}(P) \overset{?}{\leq} C.$$

Thus, the oracle can be implemented by computing P^i, a shortest s_i-t_i path, for each $i \in [k]$. If their total length is at most C then the oracle returns "Yes" and the vector $v = \sum_{i=1}^k e_{P^i}$. Otherwise it returns "No".

Lastly, recall that the efficiency of these methods heavily depends on the width. Before analyzing the width, let us make a simple observation. In any solution to the congestion minimization problem each edge can be traversed at most k times, so we may assume without loss of generality that $C \leq k$.

Claim 27.2.1. This oracle has width parameter $\rho \leq k$.

Proof. Recall that the requirement for the width parameter ρ was given in (27.1.1). In our current scenario, we will let v denote the output of the oracle, so the condition becomes

$$\left| \sum_{i \in [k]} \sum_{\substack{P \in \mathcal{P}^i \\ e \in P}} v_P^i - C \right| \leq \rho \qquad \forall e \in E.$$

As observed above, we may assume that $C \leq k$. On the other hand, $\sum_{P \in \mathcal{P}^i \,:\, e \in P} v_P^i \leq 1$ since v^i lies in the simplex $\mathcal{C}^i$. Thus, we also have $\sum_{i \in [k]} \sum_{P \in \mathcal{P}^i \,:\, e \in P} v_P^i \leq k$. $\square$

27.3 Bandits

The bandits scenario is similar to the original experts scenario, except that the algorithm only receives *partial feedback*. That is, in round t, the algorithm does not receive the entire vector c_t. Instead, if it randomly chose to follow expert j, it only receives the cost $c_{t,j}$.

27.3.1 The bandit model

There is an agent who must make a sequence of online decisions. There are a sequence of discrete time steps; in each time step, the agent must make a decision and incur the cost of that decision. The agent has limited or no forehand knowledge of what that cost will be. Of course, the goal is to incur as little cost as possible. At the end of each time step, the agent is told the cost of the choice he actually made, but not the other choices he could have made.

A typical setup is that there are n ***actions***. The costs at time t are specified by a vector $c_t = (c_{t,1}, \ldots, c_{t,n}) \in \mathbb{R}^n$, where $c_{t,j}$ is the cost of performing action j at time t. The agent must (without knowing c_t) pick an action j to perform, then incur the cost $c_{t,j}$. The cost vector c_t is **not** revealed to the agent; only the **single** cost $c_{t,j}$ is revealed.

There are many different models of bandits. Two of the most basic are ***stochastic bandits*** and ***adversarial bandits*** (or ***non-stochastic bandits***), which we discuss in this section.

"Adversarial" Bandits. This setting is most similar to the model of prediction with expert advice, as discussed in Chapter 26. Let us now consider the power of the adversary. As mentioned last time, if c_t depends on the action chosen at time t, the results would be uninteresting: the lower bound of Section 26.3.1 implies a regret of $\Omega(T)$ after T rounds.

So suppose we only allowed the adversary to determine c_t based upon the algorithm's decisions in rounds 1 through $t-1$. The adversary does not know any random bits used by the algorithm. This is called an ***adaptive online adversary***. Important results have been proven in this setting, even in the bandit model, but there is also a sentiment[1] that regret bounds against adaptive online adversaries are "not meaningful".

Today, we will consider adversarial bandits with an ***oblivious adversary***: the adversary knows the algorithm (but not its random bits), and all cost functions c_t are chosen *before* the algorithm begins execution. Already this is quite interesting because it is a substantial generalization of stochastic bandits.

[1]See Arora, Dekel, Tewari for further discussion.

As before, let A_T be the cost incurred by the algorithm up to round T. We now define

$$\operatorname{Regret}(T) \;=\; \max_{j\in[n]} \mathrm{E}\left[\, A_T - \sum_{t=1}^{T} c_{t,j} \,\right] \;=\; \mathrm{E}\,[\, A_T\,] - \min_{j\in[n]} \sum_{t=1}^{T} c_{t,j}.$$

Usually, the goal is to get $\operatorname{Regret}(T) = o(T)$.

27.3.2 Exp3

In this section we present the Exp3 algorithm, which achieves regret bound $O(\sqrt{Tn\log n})$ in the adversarial bandits setting, which is nearly optimal. The algorithm's name stands for "Exponential-weight algorithm for Exploration and Exploitation". It was original presented by Auer, Cesa-Bianchi, Freund and Schapire in the FOCS 1995 conference (Auer et al., 1995, 2002).

Overview of ideas. At a very vague level, Exp3 can be explained as follows:

Gradient Descent	is to	Random Coordinate Descent
	as	
Randomized Weighted Majority	is to	Exp3

The key idea for Exp3 is to run the Randomized Weighted Majority algorithm as a subroutine. However, recall that RWM requires a cost for all experts, whereas the bandit model only provides us the cost of a single expert. The Exp3 algorithm will create a random vector of "simulated costs" that equals the true costs in expectation, then provided that simulated vector to RWM.

Algorithm 27.2 The Exp3 algorithm. We assume each weight vector $c_t \in [0,1]^n$

```
1: procedure Exp3($\eta$)
2:     Let $y_1 = (1,\ldots,1)$
3:     for $t \leftarrow 1,2,\ldots,T$
4:         Let $x_t = y_t / \|y_t\|_1$ (normalize the weights).
5:         Use fresh randomness to pick action $X_t \in [n]$ according to distribution $x_t$.
6:         Perform the action $X_t$
7:         ▷ Expected cost incurred is $\sum_{i=1}^n c_{t,i}x_{t,i} = \langle\, c_t,\, x_t \,\rangle$
8:         ▷ Receive the single cost $c_{t,X_t} \in [0,1]$
9:         Construct the simulated cost vector $\hat{c}_t \in \mathbb{R}^n$ with
               $$\hat{c}_{t,i} \;=\; \begin{cases} c_{t,i}/x_{t,i} & \text{(for } i = X_t\text{)} \\ 0 & \text{(otherwise)} \end{cases}$$
10:        for $i \leftarrow 1,\ldots,n$
11:            $y_{t+1,i} = y_{t,i}\exp(-\eta\hat{c}_{t,i})$    ▷ (decrease weight according to action $i$'s simulated cost)
```

Let $c_t \in [0,1]^n$ denote the true cost of the actions in round t. Let F_t denote the **full history** up to the end of time t, by which we mean all random variables that have been generated up to the end of time t. See Section 29.2 for further explanation.

The following claim says that, no matter what the algorithm has learned so far, in round t the simulated costs equal the actual costs in expectation.

Claim 27.3.1.

$$\mathrm{E}\left[\,\hat{c}_{t,i} \mid F_{t-1}\,\right] = c_{t,i} \quad \forall t \in [T],\ i \in [n].$$

Proof. Recalling the definition of $\hat{c}$, we have the expression

$$\mathrm{E}\left[\,\hat{c}_{t,i} \mid F_{t-1}\,\right] \;=\; \mathrm{E}\left[\frac{c_{t,i}}{x_{t,i}} \cdot \mathbf{1}_{X_t=i} + 0 \cdot \mathbf{1}_{X_t \neq i} \;\middle|\; F_{t-1}\right]$$

Here $c_{t,i}$ is a constant, and $x_{t,i}$ is completely determined by F_{t-1}. Thus, we can use Claim 29.2.5 to pull out $c_{t,i}/x_{t,i}$, which yields

$$\begin{aligned}
&= \frac{c_{t,i}}{x_{t,i}} \cdot \mathrm{E}\left[\,\mathbf{1}_{X_t=i} \mid F_{t-1}\,\right] \\
&= \frac{c_{t,i}}{x_{t,i}} \cdot x_{t,i} \quad \text{(by the conditional distribution of } X_t\text{)} \\
&= c_{t,i}.
\end{aligned}$$

□

We will apply the Randomized Weighted Majority analysis of Theorem 26.4.4, with the simulated cost $\hat{c}_t$ rather than the actual costs c_t. That will introduce a quadratic term in the costs, which we first show how to deal with.

Claim 27.3.2.

$$\mathrm{E}\left[\sum_{i=1}^{n} x_{t,i}\hat{c}_{t,i}^2 \;\middle|\; F_{t-1}\right] \leq n \quad \forall t \in [T].$$

Proof. As in the previous proof, we use the definition of $\hat{c}$ to obtain

$$\mathrm{E}\left[\sum_{i=1}^{n} x_{t,i}\hat{c}_{t,i}^2 \;\middle|\; F_{t-1}\right] \;=\; \mathrm{E}\left[\sum_{i=1}^{n} x_{t,i}\left(\left(\frac{c_{t,i}}{x_{t,i}}\right)^2 \cdot \mathbf{1}_{X_t=i} + 0 \cdot \mathbf{1}_{X_t\neq i}\right) \;\middle|\; F_{t-1}\right] \quad \text{(definition of } \hat{c}_t\text{)}$$

Recall that $x_{t,i}\left(\frac{c_{t,i}}{x_{t,i}}\right)^2$ is completely determined by F_{t-1}. Thus, using Claim 29.2.5 and linearity of expectation, we get

$$\begin{aligned}
&= \sum_{i=1}^{n} x_{t,i}\left(\frac{c_{t,i}}{x_{t,i}}\right)^2 \cdot \mathrm{E}\left[\,\mathbf{1}_{X_t=i} \mid F_{t-1}\,\right] \\
&= \sum_{i=1}^{n} x_{t,i}\left(\frac{c_{t,i}}{x_{t,i}}\right)^2 \cdot x_{t,i} \quad \text{(by the conditional distribution of } X_t\text{)} \\
&= \sum_{i=1}^{n} c_{t,i}^2 \\
&\leq n.
\end{aligned}$$

This last inequality holds because every cost $c_{t,i}$ is at most 1. □

Theorem 27.3.3. Let $j \in [n]$ be arbitrary. Let $\eta = \sqrt{2\ln(n)/Tn}$. The expected cost of the algorithm satisfies

$$\mathrm{E}\left[\sum_{t=1}^{T} \langle\, c_t,\, x_t \,\rangle\right] - \sum_{t=1}^{T} c_{t,j} \;\leq\; \sqrt{2Tn\ln n}.$$

References: See (Hazan, 2015, Lemma 6.2).

Proof. We will apply Theorem 26.4.4 to the simulated cost vectors $\hat{c}_t$. Note that these cost vectors $\hat{c}_t$ are random (because $\hat{c}_t$ depends on X_t), but that does not matter — the analysis of Theorem 26.4.4 works for all cost vectors, even adversarially chosen ones. Thus, with probability 1,

$$\sum_{t=1}^{T}(\langle\, \hat{c}_t,\, x_t \,\rangle - \hat{c}_{t,j}) \;\leq\; \frac{\ln n}{\eta} + \frac{\eta}{2}\sum_{t=1}^{T}\sum_{i=1}^{n} x_{t,i}\hat{c}_{t,i}^2. \tag{27.3.1}$$

Now consider the t^{th} term on the left-hand side, and take its expectation conditioned on F_{t-1}. We obtain

$$\begin{aligned}
\mathrm{E}\left[\,\langle\, \hat{c}_t,\, x_t \,\rangle - \hat{c}_{t,j} \;\middle|\; F_{t-1}\,\right] \;&=\; \mathrm{E}\left[\sum_{i=1}^{n}\hat{c}_{t,i}x_{t,i} - \hat{c}_{t,j} \;\middle|\; F_{t-1}\right] \\
&=\; \sum_{i=1}^{n}\mathrm{E}\left[\,\hat{c}_{t,i}x_{t,i} \;\middle|\; F_{t-1}\,\right] - \mathrm{E}\left[\,\hat{c}_{t,j} \;\middle|\; F_{t-1}\,\right] \quad \text{(by linearity of expectation)}
\end{aligned}$$

Since x_t is completely determined by F_{t-1}, we can use Claim 29.2.5 to pull out $x_{t,i}$, which yields

$$\begin{aligned}
&=\; \sum_{i=1}^{n} x_{t,i}\,\mathrm{E}\left[\,\hat{c}_{t,i} \;\middle|\; F_{t-1}\,\right] - \mathrm{E}\left[\,\hat{c}_{t,j} \;\middle|\; F_{t-1}\,\right] \\
&=\; \sum_{i=1}^{n} x_{t,i}c_{t,i} - c_{t,j} \quad \text{(by Claim 27.3.1)} \\
&=\; \langle\, x_t,\, c_t \,\rangle - c_{t,j}.
\end{aligned} \tag{27.3.2}$$

Summing (27.3.2) over t and taking the unconditional expectation, we obtain

$$\begin{aligned}
\mathrm{E}\left[\sum_{t=1}^{T}\langle\, x_t,\, c_t \,\rangle - c_{t,j}\right] \;&=\; \mathrm{E}\left[\sum_{t=1}^{T}\mathrm{E}\left[\,\langle\, \hat{c}_t,\, x_t \,\rangle - \hat{c}_{t,j} \;\middle|\; F_{t-1}\,\right]\right] \\
&=\; \mathrm{E}\left[\sum_{t=1}^{T}(\langle\, \hat{c}_t,\, x_t \,\rangle - \hat{c}_{t,j})\right] \quad \text{(by Claim 29.2.3)} \\
&\leq\; \frac{\ln n}{\eta} + \frac{\eta}{2}\,\mathrm{E}\left[\sum_{t=1}^{T}\sum_{i=1}^{n} x_{t,i}\hat{c}_{t,i}^2\right] \quad \text{(by the expectation of (27.3.1))}
\end{aligned} \tag{27.3.3}$$

The right-hand side may be bounded as follows.

$$\begin{aligned}
\mathrm{E}\left[\sum_{t=1}^{T}\sum_{i=1}^{n} x_{t,i}\hat{c}_{t,i}^2\right] \;&=\; \mathrm{E}\left[\sum_{t=1}^{T}\mathrm{E}\left[\sum_{i=1}^{n} x_{t,i}\hat{c}_{t,i}^2 \;\middle|\; F_{t-1}\right]\right] \quad \text{(by Claim 29.2.3)} \\
&\leq\; \mathrm{E}\left[\sum_{t=1}^{T} n\right] \;=\; Tn \quad \text{(by Claim 27.3.2)}
\end{aligned} \tag{27.3.4}$$

Thus, combining (27.3.3) and (27.3.4), we obtain

$$\mathrm{E}\left[\sum_{t=1}^{T}\langle\, x_t,\, c_t \,\rangle - c_{t,j}\right] \;\leq\; \frac{\ln n}{\eta} + \frac{\eta}{2}Tn.$$

Since $\eta = \sqrt{2\ln(n)/Tn}$, this proves the theorem. ☐

Exercises

Exercise 27.1. Farkas' lemma is a fundamental result in the theory of linear inequalities. (For example, it can be used to prove the strong duality theorem for linear programming.) One statement of the lemma is as follows.

Let A be an $m \times n$ matrix and $b \in \mathbb{R}^m$. Then exactly one of the following must hold.

- **(a)** there exists $v \in \mathbb{R}^n$ with $Av \geq b$ and $v \geq 0$.
- **(b)** there exists $x \in \mathbb{R}^m$ with $x^\mathsf{T} A \leq 0$, $x^\mathsf{T} b > 0$ and $x \geq 0$.

Part I. Prove that (a) and (b) cannot simultaneously hold.

Hint: If (a) and (b) both hold, prove that $x^\mathsf{T}(Av - b) \geq 0$, then prove that $x^\mathsf{T}(Av - b) < 0$.

Part II. Imagine running the Algorithm 27.1 for an infinite number of parameters δ that tend to zero. If the oracle always outputs "Yes", then prove that (a) holds.

Hint: Use Lemma 27.1.4, Corollary 27.1.7 and the Bolzano-Weierstrass theorem.

Part III. If the oracle ever outputs "No", then prove that (b) holds.

Chapter 28

Polynomial Methods

In Section 15.1 we discussed the problem of testing equality of two bitstrings in a distributed setting. We solved that problem by comparing the hash values when using the ***polynomial hash*** functions of Section 14.3. That was our first taste of polynomial methods.

28.1 Polynomial Identity Testing

The ***polynomial identity testing*** problem (henceforth, PIT) can roughly be defined as follows. Given a multivariate polynomial $p(x_1, \ldots, x_n)$ with coefficients in some field $\mathbb{F}$, decide if p equals the zero polynomial. This problem might sound dull and algebraic, but it is perhaps better to think of it as very abstract and general. Many interesting non-algebraic problems, such as testing equality of strings, are related to PIT.

Let us now explain PIT in more detail. Every polynomial $p(x_1, \ldots, x_n)$ can be expanded into a sum of monomials with coefficients in $\mathbb{F}$. For example, the polynomial $p(x, y, z) = (x + 2y)(3y - z)$ can be expanded into a sum of monomials as

$$p(x, y, z) \;=\; 3xy + 6y^2 - xz - 2yz.$$

If p is expanded into a sum of monomials, and all coefficients of those monomials are zero, then p is called the ***zero polynomial***, or ***identically zero***.

Example 28.1.1. Is

$$p(x, y, z) \;=\; (x + y)^2 - (x - y)^2 - 4xy$$

identically zero?

Answer.

Yes, by trivial expansion.

Example 28.1.2. Is

$$p(x, y, z) \;=\; (x_1 - x_2)(x_1 - x_3)(x_2 - x_3) \;-\; \det \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{pmatrix}$$

identically zero?

Answer.

Yes, by the formula for a Vandermonde determinant.

Univariate polynomials are commonly classified by their degree. For a multivariate polynomial, this requires some explanation. A monomial is any expression of the form $c \cdot \prod_{i=1}^{n} x_i^{\alpha_i}$ where $c \in \mathbb{F}$ and $\alpha_1, \ldots, \alpha_n$ are non-negative integers. The ***total degree*** of this monomial is $\sum_{i=1}^{n} \alpha_i$. The total degree of a polynomial is the maximum total degree of its monomials.

We can now state the PIT question mathematically as follows. If p is a polynomial of total degree d, and we expand

$$p(x_1, \ldots, x_n) \;=\; \sum_{\substack{\alpha_1,\ldots,\alpha_n \geq 0 \\ \alpha_1+\cdots+\alpha_n \leq d}} c_{\alpha_1,\ldots,\alpha_n} \prod_{i=1}^{n} x_i^{\alpha_i},$$

then are all coefficients $c_{\alpha_1,\ldots,\alpha_n}$ equal to zero?

Computational considerations. As stated above, PIT is a mathematical question: is p equal to the zero polynomial, or isn't it? We are interested in the computational aspects of PIT. This requires discussing how the data is represented and what operations are permitted. In particular, we must clarify what it means to be "given p".

We will assume that p is represented as a "black box" which, given explicit values $x_1, \ldots, x_n$, returns the value $p(x_1, \ldots, x_n)$ in one unit of time. This model is useful because p might have an implicit representation, such as in Example 28.1.2. Nevertheless, given numeric values for x_1, x_2, x_3 we can evaluate $p(x_1, x_2, x_3)$ by computing the determinant numerically.

Complexity Status. The complexity status of PIT is quite interesting. We will show that there is a randomized algorithm to decide PIT. However, there is no known efficient deterministic algorithm to decide PIT. Furthermore, if such an algorithm existed then there would be significant consequences in complexity theory. It is perhaps fair to say that PIT is the canonical problem in RP that is not known to be in P.

28.1.1 The Schwartz-Zippel Lemma

The main tool that we will use in this chapter is the following.

Lemma 28.1.3 (Schwartz-Zippel Lemma). Let $\mathbb{F}$ be any field. Let $p(x_1, \ldots, x_n)$ be a polynomial of total degree d with coefficients in the field $\mathbb{F}$. Assume that p is not identically zero. Let $S \subseteq \mathbb{F}$ be any finite set. Then, if we pick $y_1, \ldots, y_n$ independently and uniformly from S,

$$\Pr[\; p(y_1, \ldots, y_n) = 0 \;] \;\leq\; \frac{d}{|S|}.$$

Interestingly, the right-hand side $\frac{d}{|S|}$ does not depend on n!

References: (Motwani and Raghavan, 1995, Theorem 7.2), Wikipedia.

To help understand the lemma, let p be a polynomial of total degree d over $\mathbb{R}$. Fix any set S of size $2d$, and pick each y_i uniformly and independently from S. According to the lemma, $(y_1, \ldots, y_n)$ is a root of p with probability at most $1/2$.

Question 28.1.4. Can we conclude that polynomials over $\mathbb{R}$ have finitely many roots?

Answer.

No! Consider the polynomial $p(x_1, x_2) = x_1$ which has total degree $d = 1$. It has *infinitely* many roots because setting $x_1 = 0$ and x_2 to anything gives a root. However, if we fix $S = \{0, 1\}$ and randomly choose $y_1, y_2 \in S$ then indeed $\Pr[\, p(y_1, y_2) = 0 \,] = 1/2$, so Lemma 28.1.3 is tight.

Proof of Lemma 28.1.3. We proceed by induction on n.

The base case is the case $n = 1$. Over any field $\mathbb{F}$, a univariate polynomial of degree d has at most d roots; see Fact A.2.14. So the probability that y_1 is a root is at most $d/|S|$.

Now we assume the theorem is true for polynomials with $n-1$ variables, and we prove it for those with n variables. The main idea is to obtain polynomials with fewer variables by factoring out the variable x_1 from p. Let k be the largest power of x_1 appearing in any monomial of p. We may write

$$p(x_1, \ldots, x_n) \;=\; \sum_{i=0}^{k} x_1^i \cdot q_i(x_2, \ldots, x_n).$$

By our choice of k, the polynomial q_k is not identically zero, and its total degree is at most $d - k$.

Now randomly choose the values of $y_2, \ldots, y_n \in S$ and define the event

$$\mathcal{E}_1 \;=\; \text{“}q_k(y_2, \ldots, y_n) = 0\text{”}.$$

By induction

$$\Pr[\, \mathcal{E}_1 \,] \;=\; \Pr[\; q_k(y_2, \ldots, y_n) = 0 \;] \;\leq\; \frac{d-k}{|S|}. \tag{28.1.1}$$

We have already chosen the values $y_2, \ldots, y_n \in S$. What remains is the *univariate* polynomial

$$f(x_1) \;=\; \sum_{i=0}^{k} x_1^i \cdot q_i(y_2, \ldots, y_n) \;=\; p(x_1, y_2, \ldots, y_n).$$

Conditioning on the event $\overline{\mathcal{E}_1}$, the coefficient of x_1^k in f is non-zero, and so f is not identically zero. Now choose y_1 uniformly at random from S, and define the event

$$\mathcal{E}_2 \;=\; \text{“}f(y_1) = 0\text{”} \;=\; \text{“}p(y_1, \ldots, y_n) = 0\text{”}.$$

By the univariate argument of the base case,

$$\Pr\left[\, \mathcal{E}_2 \mid \overline{\mathcal{E}_1} \,\right] \;=\; \Pr\left[\, f(y_1) = 0 \mid \overline{\mathcal{E}_1} \,\right] \;\leq\; \frac{k}{|S|}. \tag{28.1.2}$$

We can combine the two types of errors in a familiar way.

$$\begin{aligned}
\Pr[\, p(y_1, \ldots, y_n) = 0 \,] &= \Pr[\, \mathcal{E}_2 \,] \\
&\leq \Pr[\, \mathcal{E}_1 \,] + \Pr\left[\, \mathcal{E}_2 \mid \overline{\mathcal{E}_1} \,\right] && \text{(by Exercise A.4)} \\
&\leq \frac{d-k}{|S|} + \frac{k}{|S|} && \text{(by (28.1.1) and (28.1.2))} \\
&= \frac{d}{|S|} && \square
\end{aligned}$$

28.1.2 Solving PIT

Algorithm 28.1 Randomized algorithm for PIT. It is assumed that p has coefficients in $\mathbb{F}$ and $S \subseteq \mathbb{F}$.

```
1: function SolvePIT(polynomial $p(x_1, \ldots, x_n)$, set $S$)
2:     Pick $y_1, \ldots, y_n$ uniformly and independently from $S$
3:     if $p(y_1, \ldots, y_n) = 0$ then return True
4:     else return False
5: end function
```

Recall the definitions of an ***RP-algorithm*** and a ***coRP-algorithm*** from Section 1.2.

Theorem 28.1.5. Let d be the total degree of p. If $|S| \geq 2d$ then SolvePIT is a coRP algorithm for the PIT problem.

Proof. If the correct output is True then p is identically zero, so $p(y_1, \ldots, y_n)$ will always equal 0, and the algorithm will always output True. So the algorithm has no false negatives.

If the correct output is False then the algorithm will incorrectly output True if $p(y_1, \ldots, y_n) = 0$. By Lemma 28.1.3, this happens with probability at most $d/|S| \leq 1/2$. ☐

28.2 Bipartite Matching

Let $G = (U \cup V, E)$ be a bipartite graph, meaning that U and V are disjoint sets of vertices, and every edge in E has exactly one endpoint in U and exactly one endpoint in V. A ***matching*** in G is a set of edges that share no endpoints. A ***perfect matching*** in G is a set of edges $M \subseteq E$ such that every vertex is contained in exactly one edge of M.

Polynomial time algorithms are known to decide if G has a perfect matching, and even to construct such a matching. We will give a randomized algorithm to decide if G has a perfect matching, by reducing that problem to PIT.

Let A be the matrix whose rows are indexed by the vertices in U and columns are indexed by the vertices in V. The entries of A are:

$$A_{u,v} = \begin{cases} x_{u,v} & (\text{if } uv \in E) \\ 0 & (\text{if } uv \notin E) \end{cases}, \tag{28.2.1}$$

where $\{\, x_{u,v} \,:\, uv \in E \,\}$ are distinct variables.

Another consequence of Claim 28.2.1 is that $\det A$ can be viewed as a polynomial over *any* field $\mathbb{F}$. This is because all the monomials have coefficient either $+1$ or -1, which are numbers in every field $\mathbb{F}$.

Claim 28.2.1.

$$\det A = \sum_{\text{perfect matching } m:U \to V} \pm \prod_{u \in U} x_{u,m(u)}. \tag{28.2.2}$$

Corollary 28.2.2. $\det A$ is identically zero if and only if G has no perfect matching.

Proof of Claim 28.2.1. By the Leibniz formula for determinants,

$$\det A = \sum_{\text{bijection } \pi:U \to V} \pm \prod_{u \in U} A_{u,\pi(u)}, \tag{28.2.3}$$

where the signs are irrelevant for our purposes.

The first observation is that the monomials in the right-hand side of (28.2.3) all involve distinct sets of variables, and therefore there can be no cancellations amongst these monomials.

Next, observe that a *bijection* from U to V is simply a pairing $\{ (u, \pi(u)) : u \in U \}$ of elements in U and elements in V such that each vertex appears in exactly one pair. In contrast, a *perfect matching* is such a bijection in which every pair $(u, \pi(u))$ is also an edge in E.

Lastly, observe that a monomial $\prod_{u \in U} A_{u,\pi(u)}$ is non-zero if and only if every pair $(u, \pi(u))$ is an edge in E. This is because $A_{u,\pi(u)} = 0$ if there is no edge from u to $\pi(u)$. It follows that the non-zero summands in (28.2.3) are precisely the summands in (28.2.2). $\square$

28.2.1 A perfect matching algorithm

Algorithm 28.2 Randomized algorithm for deciding if G has a perfect matching. Here $G = (U, V, E)$ is a bipartite graph and $n = |U| = |V|$.

```
1: function DecidePM(graph $G$)
2:     Let $\mathbb{F}$ be any field and $S$ any set where $S \subseteq \mathbb{F}$ and $|S| \geq 2n$
3:     Let $A$ be the matrix described in (28.2.1)
4:     Replace every non-zero entry of $A$ with an independent random number uniform in $S$
5:     if $\det A \neq 0$ then return True else return False
6: end function
```

Question 28.2.3. What is the running time of this algorithm?

Answer.

Assume that operations in $\mathbb{F}$ take $O(1)$ time. Using Gaussian elimination, one can compute $\det A$ in time $O(n^3)$. This can be improved to $O(n^{2.38})$ by a galactic algorithm.

Theorem 28.2.4. DecidePM is an RP-algorithm for deciding if G has a perfect matching.

Proof. After line 3, $\det A$ is a polynomial of total degree at most n. As observed above, it can be viewed as a polynomial over any field $\mathbb{F}$.

If G has no perfect matching then by Corollary 28.2.2, $\det A$ is identically zero and so the algorithm will always return False. Therefore there are no false positives.

If G does have a perfect matching then, by Corollary 28.2.2, $\det A$ is not identically zero. By the Schwartz-Zippel lemma (Lemma 28.1.3), the false negative probability is at most

$$\frac{\text{total degree}}{|S|} \leq \frac{n}{2n} = \frac{1}{2}. \qquad \square$$

Question 28.2.5. How can we decrease the failure probability to δ?

Answer.

One approach is to use probability amplification by repeated trials, as in Chapter ??. Another approach is to increase S (and $\mathbb{F}$) to have size at least n/δ.

28.3 Exercises

Exercise 28.1. Let $a = (a_1, \ldots, a_s)$ be a string of characters in $[\![q]\!]$, Let p be a prime with $p \geq q$. Let

$$h(a) \;=\; \Big(\sum_{i=1}^{s} a_i X_i + Y \Big) \bmod p.$$

be a linear hash function as defined in (14.1.1). Use Lemma 28.1.3 to prove that collisions are unlikely:

$$\Pr\left[\, h(a) = h(b) \,\right] \;\leq\; \frac{1}{p}.$$

This is similar to Corollary 14.1.8.

Exercise 28.2. Give a coRP algorithm for the PIT problem under the assumption that $|\mathbb{F}| > d$.

Exercise 28.3. Give an example of a field $\mathbb{F}$ and a polynomial $p(x_1, \ldots, x_n)$ such that p is *not* the zero polynomial, but nevertheless $p(x_1, \ldots, x_n) = 0$ for all $x_1, \ldots, x_n \in \mathbb{F}$.

Conclude that, without the assumption $|\mathbb{F}| > d$, it is *impossible* to solve PIT when p is represented as a black box.

Exercise 28.4. Let $p(x_1, \ldots, x_n)$ be a polynomial over $\mathbb{F}_2$, represented explicitly with a formula. Prove that deciding whether there exist $x_1, \ldots, x_n \in \mathbb{F}_2$ such that $p(x_1, \ldots, x_n) \neq 0$ is NP-hard.

Exercise 28.5. Suppose that $p(x_1, \ldots, x_n)$ is represented as an explicit formula (using addition, subtraction, multiplication and parentheses). Let d be the total degree of p.

If $d \geq |\mathbb{F}|$, then Lemma 28.1.3 does not give any useful information about the number of roots. Despite that, give a coRP algorithm for PIT.

Exercise 28.6. Algorithm 28.2 is a randomized, polynomial time algorithm for deciding if a bipartite graph has a *perfect* matching.

Part I. Give a randomized, polynomial time algorithm for deciding if a bipartite graph has a matching of size at least k. You should use or modify Algorithm 28.2.

Note: k can depend on n.

Part II. BONUS: Give a randomized algorithm to find the size of a maximum matching in $O(n^{2.38})$ time.

Exercise 28.7. Let A, B, C be $n \times n$ matrices with entries in any field $\mathbb{F}$. We would like to decide if $A \cdot B \overset{?}{=} C$. Of course this can be solved exactly in $O(n^3)$ time by matrix multiplication. The following interesting algorithm has runtime only $O(n^2)$.

Algorithm 28.3 An algorithm for deciding if $A \cdot B \overset{?}{=} C$. Assume that $\mathbb{F}$ is a finite field.

```
1: function MatMult(matrix $A, B, C$)
2:     Pick vectors $r$ and $s$ in $\mathbb{F}^n$ uniformly at random
3:     if $(r^\mathsf{T} A) \cdot (Bs) = r^\mathsf{T} C s$ then return True
4:     else return False
5: end function
```

Part I. Define a polynomial p with $2n$ variables such that $A \cdot B \neq C$ if and only if p is not identically zero.

Part II. Assuming that $\mathbb{F}$ is finite with $|\mathbb{F}| > 2$, prove that MatMult can decide if $A \cdot B \overset{?}{=} C$ with failure probability at most $3/4$.

Part III. How can the algorithm be modified to handle the case $\mathbb{F} = \mathbb{R}$?

Part IV. BONUS: Even if $|\mathbb{F}| = 2$, show that the failure probability is still at most $3/4$.

Chapter 29

Martingales

Probability begins with the definition of independence.
"Probability: Theory and Examples", R. Durrett

29.1 Introduction

When one is faced with multiple random variables, perhaps the first question one should ask is whether they are independent. If so, this may simplify their analysis because they can be analyzed separately. Moreover, it is simpler to *generate* independent random variables; for example, they can be generated simultaneously, possibly even on different computers.

Computer programs running on a single machine have a temporal aspect that is not well captured by the notion of independence. For example, a program might generate at each time step a "new" random number, but one whose distribution somehow depends on the previous random numbers.

One common model for sequences of dependent random variables is a ***Markov chain***. These are typically used to model random sequences of "states" (categorical RVs), in which the distribution of the next state is determined once the current state is known.

References: (Anderson et al., 2017, Definition 10.42), (Motwani and Raghavan, 1995, Section 6.2), (Mitzenmacher and Upfal, 2005, Section 7.1), (Grimmett and Stirzaker, 2001, Section 6.1), Wikipedia.

In contrast, ***martingales*** are sequences of *real-valued* RVs in which the next RV can depend arbitrarily on all previous RVs; however, they must satisfy

the expectation of the next RV, conditioned on all previous RVs, must equal the current RV.

This is of course a vague description, but it conveys the key ideas and allows us to discuss some examples.

Question 29.1.1. Let $S_0, S_1, \ldots, S_T \in \{\textsc{Sunny}, \textsc{Cloudy}\}$ be random variables satisfying

$$\begin{aligned}
\Pr[\, S_0 = \textsc{Sunny} \,] &= 1/2 \\
\Pr[\, S_0 = \textsc{Cloudy} \,] &= 1/2 \\
\Pr[\, S_t = S_{t-1} \,] &= 2/3 \qquad \forall t \in [T] \\
\Pr[\, S_t \neq S_{t-1} \,] &= 1/3 \qquad \forall t \in [T].
\end{aligned}$$

Are these a Markov chain or a martingale?

Answer.

These are categorical random variables, so they cannot be a martingale. They are a Markov chain because the distribution of S_t is completely determined by the value of S_{t-1}.

Example 29.1.2. Let $S_0, S_1, \ldots, S_T \in \mathbb{Z}$ satisfy

$$\begin{aligned}
\Pr[\, S_0 = 1 \,] &= 1/2 \\
\Pr[\, S_0 = 2 \,] &= 1/2 \\
\Pr[\, S_t - S_{t-1} = S_0 \,] &= 1/2 \quad \forall t \in [T] \\
\Pr[\, S_t - S_{t-1} = -S_0 \,] &= 1/2 \quad \forall t \in [T]
\end{aligned}$$

Are these a Markov chain or a martingale?

These are a martingale because

$$\mathrm{E}[\, S_t \mid S_0, \ldots, S_{t-1} \,] \;=\; \frac{1}{2} \cdot (S_{t-1} + S_0) + \frac{1}{2} \cdot (S_{t-1} - S_0) \;=\; S_{t-1}.$$

However, they are not a Markov chain. For example, suppose that $S_1 = 0$. Then S_2 might take values ± 1 with probability $1/2$ (if $S_0 = 1$), or might take values ± 2 with probability $1/2$ (if $S_0 = 2$). Since the distribution cannot be determined from S_1 alone, they do not form a Markov chain.

Question 29.1.3. Let $S_0, S_1, \ldots, S_T \in \mathbb{Z}$ be a ***standard random walk***, meaning that $S_0 = 0$ and

$$\Pr[\, S_t - S_{t-1} = \pm 1 \,] \;=\; 1/2 \quad \forall t \in [T]$$

Are these a Markov chain or a martingale?

Answer.

They are both! They are a Markov chain because the distribution of S_t depends only on S_{t-1}. They are a martingale because

$$\mathrm{E}[\, S_t \mid S_0, \ldots, S_{t-1} \,] \;=\; \frac{1}{2} \cdot (S_{t-1} + 1) + \frac{1}{2} \cdot (S_{t-1} - 1) \;=\; S_{t-1}.$$

29.2 Definitions

Consider a randomized algorithm that runs at discrete time steps $t \in [T]$. We will keep track of the ***full history***[1] of its random variables as follows.

$$\text{Full history:} \quad F_t \;=\; (\text{all random variables that have been generated by the end of time } t) \quad \forall t \in [T].$$

As more RVs are generated over time, we have the obvious[2] containment

$$F_0 \;\subseteq\; F_1 \;\subseteq\; F_2 \;\subseteq\; \cdots \;\subseteq\; F_T.$$

Typically we set $F_0 = \emptyset$ for notational convenience.

With respect to this full history, a ***martingale*** is a sequence of random variables

$$S_0,\; S_1,\; S_2,\; \ldots,\; S_T$$

satisfying three conditions.

[1] A reader comfortable with measure theoretic probability will notice that this concept (and its notation) is building toward the idea of a filtration (Grimmett and Stirzaker, 2001, page 473) (Klenke, 2008, Definition 9.9).

[2] We will use the notation of sets for operations on F_t, although strictly speaking they should be viewed as vector-valued random variables.

- To rule out pathological situations, we will require that $\mathrm{E}[\,|S_t|\,]$ is finite for all t.

- The value of S_t is completely determined by time t. In other words, the value of S_t is completely determined by the vector F_t.

- Given everything that is known at time $t-1$, we expect that S_t will equal S_{t-1}. In symbols,

$$\mathrm{E}[\,S_t \mid F_{t-1}\,] \;=\; S_{t-1}. \tag{29.2.1}$$

References: (Motwani and Raghavan, 1995, Definition 4.11), (Grimmett and Stirzaker, 2001, Definition 12.1.8), (Roch, 2020, Definition 3.30), (Durrett, 2019, Section 4.2), (Klenke, 2008, Definition 9.24), Wikipedia.

The theory of martingales becomes much richer if we consider *infinite* sequences of random variables. However in computer science applications that is not always necessary, and we can illustrate many of the key ideas by focusing on the case of finite[3] sequences.

29.2.1 Background on expectations

In order to understand martingales, it is important to have a solid understanding of conditional expectations. Defining conditional expectations for continuous RVs requires considerable care[4], so we will assume for simplicity that all RVs are discrete. Let us first review the definitions.

Definition 29.2.1 (Expectation conditioned on an event). Without loss of generality we consider events of the form "$B = b$", where B is a RV and b is a fixed constant. The conditional expectation $\mathrm{E}[\,A \mid B=b\,]$ is a non-random scalar. It is defined to equal

$$\mathrm{E}[\,A \mid B=b\,] \;=\; \sum_a a \cdot \Pr[\,A=a \mid B=b\,]. \tag{29.2.2}$$

References: (Lehman et al., 2018, Definition 19.4.5), (Anderson et al., 2017, Definitions 10.2 and 10.7).

Definition 29.2.2 (Expectation conditioned on a RV). $\mathrm{E}[\,A \mid B\,]$ is a ***random variable*** that takes the value $\mathrm{E}[\,A \mid B=b\,]$ whenever $B=b$.

References: (Anderson et al., 2017, Definition 10.23), (Mitzenmacher and Upfal, 2005, Definition 2.7), (Motwani and Raghavan, 1995, Definition 4.4), (Grimmett and Stirzaker, 2001, Definition 3.7.3).

Expectation of conditional expectation. Since $\mathrm{E}[\,A \mid B\,]$ is a random variable, we can take its expectation. This amounts to eliminating the conditioning on B.

Claim 29.2.3 (Law of total expectation). $\mathrm{E}[\,\mathrm{E}[\,A \mid B\,]\,] = \mathrm{E}[\,A\,]$.

[3] In particular, this restricts our attention to *uniformly integrable martingales/Doob martingales*. See, e.g., (Grimmett and Stirzaker, 2001, Example 12.3.9), (Durrett, 2019, Section 4.6), (Klenke, 2008, Theorem 11.7).

[4] See, e.g., (Grimmett and Stirzaker, 2001, Section 10.2 and 10.4), (Grimmett and Stirzaker, 2001, Section 4.6), (Durrett, 2019, Section 4.1), (Klenke, 2008, Chapter 8).

Proof.

$$\begin{aligned}
\mathrm{E}[\,\mathrm{E}[\,A \mid B\,]\,] &= \sum_b \mathrm{E}[\,A \mid B=b\,] \cdot \Pr[\,B=b\,] && \text{(by Definition A.3.9)}\\
&= \sum_b \Big(\sum_a a \cdot \Pr[\,A=a \mid B=b\,] \Big) \cdot \Pr[\,B=b\,] && \text{(by (29.2.2))}\\
&= \sum_a a \cdot \Big(\sum_b \Pr[\,A=a \wedge B=b\,] \Big) && \text{(by Definition A.3.4)}\\
&= \sum_a a \cdot \Pr[\,A=a\,] && \text{(by Fact A.3.6)}\\
&= \mathrm{E}[\,A\,]. \quad \square
\end{aligned}$$

References: (Anderson et al., 2017, Equation (10.28) and (10.29)), (Mitzenmacher and Upfal, 2005, Theorem 2.7), (Grimmett and Stirzaker, 2001, Theorem 3.7.4), (Motwani and Raghavan, 1995, Lemma 4.9), (McDiarmid, 1998, equation (31)), Wikipedia.

In fact, the same argument works even if all expectations are conditioned on C.

Claim 29.2.4 (Tower property). $\mathrm{E}[\,\mathrm{E}[\,A \mid B, C\,] \mid C\,] = \mathrm{E}[\,A \mid C\,]$.

References: (Motwani and Raghavan, 1995, Lemma 4.14), (Durrett, 2019, Theorem 4.1.13(ii)), (Klenke, 2008, Theorem 8.14(iv)), (Grimmett and Stirzaker, 2001, Exercise 3.7.1(f)), (McDiarmid, 1998, equation (30)), Wikipedia.

With ordinary expectations, any constants can be pulled outside due to linearity of expectation. For example, $\mathrm{E}[\,aX\,] = a\,\mathrm{E}[\,X\,]$ if a is a constant. Interestingly, when using conditional expectations, it is possible to pull certain *random variables* outside of the expectation, since conditional expectations are themselves random variables.

Claim 29.2.5 (Pulling out known factors). Let A, B and C be random variables such that A is completely determined by C. (That is, $A = f(C)$ for some function f.) Then

$$\mathrm{E}[\,A \cdot B \mid C\,] \;=\; A \cdot \mathrm{E}[\,B \mid C\,].$$

References: (Anderson et al., 2017, Equation (10.36)), (Grimmett and Stirzaker, 2001, Exercise 3.7.1(e)), (Klenke, 2008, Theorem 8.14(iii)).

29.2.2 An equivalence for martingales

Martingales can be characterized as sequences of RVs that give the best guess about a future RV.

"for finite [sequences] all corresponding martingales may be obtained in this way."

Colin McDiarmid, "Concentration".

Theorem 29.2.6 (Informal).

For all t, S_t is the best guess for S_T, given everything known at time t

$\iff$ *$S_0, \ldots, S_T$ is a martingale*

To prove this we will separately consider the two directions. First we show that a martingale is completely determined by the final RV S_T, together with the sequence $F_0, \ldots, F_T$.

Claim 29.2.7. For any martingale $S_0, S_1, \ldots, S_T$,

$$S_t \;=\; \mathrm{E}[\,S_T \mid F_t\,] \qquad \forall t \le T.$$

References: (Durrett, 2019, Theorem 4.2.5), (Grimmett and Stirzaker, 2001, Exercise 12.1.2), (Klenke, 2008, Remark 9.27).

Proof. Observe that F_k, the RVs generated up to time k, can be regarded as a *pair* of RVs

$$F_k \;=\; F_{k-1},\; F_k \backslash F_{k-1}, \tag{29.2.3}$$

of which the latter describes the new RVs generated *at* time k.

The martingale property (29.2.1) says that

$$\mathrm{E}\left[\, S_T \mid F_{T-1} \,\right] \;=\; S_{T-1}. \tag{29.2.4}$$

This proves the claim for $t = T-1$.

Now take the expectation conditioned on F_{T-2}, to obtain

$$\begin{aligned}
\mathrm{E}\left[\, S_T \mid F_{T-2} \,\right] &= \mathrm{E}\left[\, \mathrm{E}\left[\, S_T \mid F_{T-2}, F_{T-1} \backslash F_{T-2} \,\right] \mid F_{T-2} \,\right] && \text{(by Claim 29.2.4)}\\
&= \mathrm{E}\left[\, \mathrm{E}\left[\, S_T \mid F_{T-1} \,\right] \mid F_{T-2} \,\right] && \text{(by (29.2.3))}\\
&= \mathrm{E}\left[\, S_{T-1} \mid F_{T-2} \,\right] && \text{(by (29.2.4))}\\
&= S_{T-2} && \text{(by (29.2.1)).}
\end{aligned}$$

This proves the claim for $t = T-2$. Iterating this argument proves the claim for all t. □

It is worth calling attention to the following special case.

Corollary 29.2.8. If $S_0, S_1, \ldots, S_T$ is a martingale then

$$\mathrm{E}\left[\, S_T \,\right] \;=\; \mathrm{E}\left[\, S_0 \,\right].$$

Moreover, if S_0 is not random (which would hold if $F_0 = \emptyset$, for example), then $\mathrm{E}\left[\, S_T \,\right] = S_0$.

References: (Mitzenmacher and Upfal, 2005, Lemma 12.1).

Proof. Simply apply Claim 29.2.7 with $t = 0$ then take the unconditional expectation. □

The following claim gives the other direction of Theorem 29.2.6.

Claim 29.2.9. Let S be any random variable with $\mathrm{E}\left[\, |S| \,\right]$ finite. Define

$$S_t \;=\; \mathrm{E}\left[\, S \mid F_t \,\right] \qquad \forall t \in \{0, \ldots, T\}.$$

Then $S_0, S_1, \ldots, S_T$ is a martingale.

The proof is left as Exercise 29.1.

References: (Motwani and Raghavan, 1995, Theorem 4.13), (Roch, 2020, Example 3.34), (Klenke, 2008, Exercise 9.2.1).

29.2.3 Relaxing to inequalities

Sometimes we encounter random variables that satisfy (29.2.1) with *in*equality instead of equality. These can still be useful, so they deserve their own names.

$$\begin{aligned}
&\text{A } \textbf{\textit{supermartingale}} \text{ satisfies:} && \mathrm{E}\left[\, S_t \mid F_{t-1} \,\right] \;\leq\; S_{t-1} && \forall t \in [T]\\
&\text{A } \textbf{\textit{submartingale}} \text{ satisfies:} && \mathrm{E}\left[\, S_t \mid F_{t-1} \,\right] \;\geq\; S_{t-1} && \forall t \in [T].
\end{aligned}$$

These names might be the opposite of what you might expect: a supermartingale goes *down* in expectation, whereas a submartingale goes *up* in expectation. Oh well — the names are firmly established, and we're stuck with them.

References: (Grimmett and Stirzaker, 2001, Definition 12.1.9), (Motwani and Raghavan, 1995, Definition 4.7), (Durrett, 2019, Section 4.2), (Roch, 2020, Definition 3.30), (Klenke, 2008, Definition 9.24), Wikipedia.

The following claim gives a trivial generalization of Corollary 29.2.8 to these settings as well.

Claim 29.2.10. Suppose that $S_0, S_1, \dots, S_T$ is:

$$\begin{aligned} \text{a } \textbf{\textit{supermartingale}}. \text{ Then} \quad & \mathrm{E}[\,S_T\,] \;\leq\; \mathrm{E}[\,S_0\,] \\ \text{a } \textbf{\textit{submartingale}}. \text{ Then} \quad & \mathrm{E}[\,S_T\,] \;\geq\; \mathrm{E}[\,S_0\,]. \end{aligned}$$

29.3 Azuma's Inequality

For our purposes, the main appeal of martingales is that they have powerful concentration properties, despite lacking independence. This is one reason that martingales find many uses in computer science.

The main martingale concentration result we will use is Azuma's inequality, which is shown in the following theorem. It is a generalization[5] of Hoeffding's inequality, which we discussed in Section 9.3 and Section 21.3.

Theorem 29.3.1. Let $S_0, S_1, \dots, S_n$ be a martingale, as in Section 29.2. Suppose that $|S_t - S_{t-1}| \leq 1$ for all $t \in [n]$. For any $\alpha \geq 0$,

$$\begin{aligned} & \Pr[\,S_n \geq S_0 + \alpha\,] && \leq \; \exp(-\alpha^2/2n) \\ \text{and} \quad & \Pr[\,S_n \leq S_0 - \alpha\,] && \leq \; \exp(-\alpha^2/2n). \\ \text{Combining these} \quad & \Pr[\,|S_n - S_0| \geq \alpha\,] && \leq \; 2\exp(-\alpha^2/2n). \end{aligned}$$

References: (McDiarmid, 1998, Theorem 3.10), (Cesa-Bianchi and Lugosi, 2006, Lemma A.7), (Motwani and Raghavan, 1995, Theorem 4.16), (Mitzenmacher and Upfal, 2005, Theorem 12.6), (Alon and Spencer, 2000, Theorem 7.2.1), (Roch, 2020, Theorem 3.52), (Grimmett and Stirzaker, 2001, Theorem 12.2.3), (Wainwright, 2019, Corollary 2.20), (Klenke, 2008, Exercise 9.2.4), Wikipedia.

29.4 Applications of Azuma's Inequality

29.4.1 Balls and bins, Bloom filters

Our first application is intended to illustrate two ideas.

- An approach to show strong concentration bounds for a RV that is *not* a sum of independent RVs.
- An argument that our best guess for a RV does not change much over time.

Consider the following algorithm that throws b balls independently into m bins. The number X of non-empty bins is returned at the end.

[5] In fact, this generalization was known to Hoeffding. In his paper, he wrote "the inequalities... remain true [with] the weaker assumption that the sequence... is a martingale."

```
function ThrowBalls(int $b$, int $m$)
    Create array $B[1..m]$ of Booleans, initially all False
    for $t = 1, \ldots, b$
        Generate $Y_t$ independently and uniformly in $[m]$        ▷ The bin for ball $t$
        Set $B[Y_t] \leftarrow$ True
    Compute $X$, the number of True entries of $B$
    return $X$
end function
```

We remark that X can be expressed as

$$X \;=\; \mathrm{NonEmpty}(Y) \;=\; |\{\, Y_t \,:\, t \in [b] \,\}|.$$

So the code needn't maintain the array B; that is just a conceptual convenience.

Our Bloom filter discussion in Section 12.4 relies on a balls and bins analysis. Specifically, when $b = 1.45m$ we showed in Claim 12.4.14 that $\mathrm{E}[\,X\,] < 0.499m$. The runtime analysis of the Bloom filter initialization relies on showing that $\Pr[\,X \geq m/2\,]$ is small. Since X is not a sum of independent RVs, we cannot use the Chernoff or Hoeffding inequalities. Nevertheless, we can give a tail bound for X using Azuma's inequality.

Theorem 29.4.1.

$$\Pr\Big[\,X - \mathrm{E}[\,X\,] \geq c\sqrt{b}\,\Big] \;\leq\; \exp(-c^2/2).$$

Following Section 29.2, we let F_t be all RVs created up to the end of iteration t of the outer loop. Then the full history is as follows.

$$\begin{aligned}
F_0 &= \emptyset \\
F_1 &= (Y_1) \\
F_2 &= (Y_1, Y_2) \\
F_3 &= (Y_1, Y_2, Y_3) \\
\vdots &= \vdots \\
F_t &= (\, Y_i \,:\, i \leq t\,) \qquad \forall t \in [b].
\end{aligned}$$

Our best guess for X changes over time as the full history unfolds. Define

$$S_t \;=\; \mathrm{E}[\,X \mid F_t\,] \qquad \forall t \in \{0, \ldots, b\}\,.$$

Then $S_0, S_1, \ldots, S_b$ is a martingale, by Claim 29.2.9.

Note that F_b contains all generated random variables. So the expectation conditioned on F_b does nothing at all — all RVs are known once F_b is known. Thus

$$S_n \;=\; \mathrm{E}[\,X \mid F_b\,] \;=\; X.$$

On the other hand F_0 contains no random variables. So the expectation conditioned on F_0 is just the *unconditional* expectation. Thus S_0 is *not* random, and simply equals

$$S_0 \;=\; \mathrm{E}[\,X \mid F_0\,] \;=\; \mathrm{E}[\,X\,]\,.$$

A key observation is that the best guess for X changes by at most 1 in each time step.

Claim 29.4.2. Fix any $t \in [b]$. Then $|S_t - S_{t-1}| \leq 1$.

Proof sketch. Recall that

$$\begin{aligned} S_{t-1} &= \mathrm{E}[\, X \mid Y_1, \ldots, Y_{t-1} \,] \\ S_t &= \mathrm{E}[\, X \mid Y_1, \ldots, Y_{t-1}, Y_t \,]. \end{aligned}$$

Choosing a location for ball t can affect the number of non-empty bins by at most 1, so $|S_t - S_{t-1}| \leq 1$. $\square$

Proof. Let $Y_1, \ldots, Y_b$ be the random choices generated by ThrowBalls(b, m). Now let $Z_1, \ldots, Z_b$ be a copy of Y where we generate a new random value for Z_t. Clearly

$$|\mathrm{NonEmpty}(Y) - \mathrm{NonEmpty}(Z)| \;\leq\; 1 \tag{29.4.1}$$

since only one ball has changed its location.

Now we will take the expectation conditioned on $F_t = \{Y_1, \ldots, Y_t\}$, but *not* conditioning on the new value Z_t. The key observation is that

$$\mathrm{E}[\, \mathrm{NonEmpty}(Z) \mid F_t \,] \;=\; \mathrm{E}[\, \mathrm{NonEmpty}(Y) \mid F_{t-1} \,]. \tag{29.4.2}$$

This is because Z has the same distribution as Y, and we have *not* conditioned on Z_t. We get

$$\begin{aligned} 1 &\geq \mathrm{E}[\, |\mathrm{NonEmpty}(Y) - \mathrm{NonEmpty}(Z)| \mid F_t \,] && \text{(by (29.4.1))} \\ &\geq |\, \mathrm{E}[\, \mathrm{NonEmpty}(Y) \mid F_t \,] - \mathrm{E}[\, \mathrm{NonEmpty}(Z) \mid F_t \,] \,| && \text{(by Jensen's inequality, Fact B.4.2)} \\ &= |\mathrm{E}[\, \mathrm{NonEmpty}(Y) \mid F_t \,] - \mathrm{E}[\, \mathrm{NonEmpty}(Y) \mid F_{t-1} \,]| && \text{(by (29.4.2))} \\ &= |S_t - S_{t-1}|. \quad \square \end{aligned}$$

Claim 29.4.2 shows that the hypotheses of Azuma's inequality (Theorem 29.3.1) are satisfied. We obtain

$$\Pr\left[X - \mathrm{E}[\, X \,] \geq c\sqrt{b} \right] \;=\; \Pr\left[S_b - S_0 \geq c\sqrt{b} \right] \;\leq\; \exp(-c^2/2).$$

This proves Theorem 29.4.1.

29.4.2 Vertex coloring

Our next application is intended to illustrate two ideas.

- It can be useful for RVs to enter the full history in a way that is not the most obvious.
- It is possible to show that a RV is concentrated around some value, without knowing what that value is!

A ***vertex coloring*** of a graph $G = (V, E)$ is a function $f : V \to [c]$ such that $f(u) \neq f(v)$ for all $uv \in E$. The ***chromatic number*** of G is the minimum integer c for which a coloring exists. This is often denoted $\chi(G)$.

Consider the following algorithm for computing the chromatic number of a random graph. The algorithm is not interesting, it just helps to understand the temporal aspect of the random variables.

Algorithm 29.1 An algorithm that generates a random graph in which each edge is added independently with probability p, then computes its chromatic number.

```
1: function RandGraph(int $n$, float $p$)
2:     Create graph $G = (V, E)$ with $V \leftarrow [n]$ and $E \leftarrow \emptyset$
3:     for $t = 1, \ldots, n$
4:         for $s = 1, \ldots, t-1$
5:             Add edge $st$ to $E$ with probability $p$
6:     Compute $\chi(G)$ by exhaustive search
7:     return $\chi(G)$
8: end function
```

Our main result shows that $\chi(G)$ is tightly concentrated around its expectation. Mysteriously, the theorem does not reveal what that expectation is.

Theorem 29.4.3.

$$\Pr\left[\,|\chi(G) - \mathrm{E}[\,\chi(G)\,]| \geq c\sqrt{n}\,\right] \;\leq\; 2\exp(-c^2/2).$$

References: (Alon and Spencer, 2000, Theorem 7.2.4), (Mitzenmacher and Upfal, 2005, Section 12.5.4), (Motwani and Raghavan, 1995, Exercise 4.11).

Following Section 29.2, we let F_t be all RVs created up to the end of iteration t of the outer loop. Let $X_{uv} \in \{0, 1\}$ be the RV that indicates if edge $uv \in E$. Then the full history is as follows.

$$\begin{aligned}
F_0 &= \emptyset \\
F_1 &= \emptyset \qquad \text{(no edges are added when } t = 1) \\
F_2 &= (X_{12}) \\
F_3 &= (X_{12}, X_{13}, X_{23}) \\
\vdots &= \vdots \\
F_t &= (\, X_{uv} \,:\, 1 \leq u < v \leq t\,) \qquad \forall t \in [n].
\end{aligned}$$

Our best guess for $\chi(G)$ changes over time as the full history unfolds. Define

$$S_t \;=\; \mathrm{E}[\,\chi(G) \mid F_t\,] \qquad \forall t \in \{0, \ldots, n\}\,.$$

Then $S_0, S_1, \ldots, S_n$ is a martingale, by Claim 29.2.9. Note that S_n (which equals $\chi(G)$) is a RV giving the chromatic number of a random graph with parameters n and p. On the other hand S_0 (which equals S_1) is *not* random and simply equals $\mathrm{E}[\,\chi(G)\,]$.

The key is to understand how much the best guess for $\chi(G)$ changes at each time step.

Claim 29.4.4. Fix any $t \in [n]$. Then $|S_t - S_{t-1}| \leq 1$.

Proof sketch. Recall that

$$\begin{aligned}
S_{t-1} &= \mathrm{E}[\,\chi(G) \mid F_{t-1}\,] \\
S_t &= \mathrm{E}[\,\chi(G) \mid F_t\,]\,.
\end{aligned}$$

Choosing the edges for vertex t affects the chromatic number by at most one, because we can always give t a new color. □

Proof. Let G be the random graph generated by $\textsc{RandGraph}(n,p)$. Now let H be a copy of G where we generate fresh independent samples for all edges in $E_t = \{\, st \,:\, s<t \,\}$. Observe that

$$|\chi(G) - \chi(H)| \;\leq\; 1. \tag{29.4.3}$$

This is because any coloring of G can be modified to a coloring of H by giving vertex t a new color, or vice versa. Now we take the expectation of (29.4.3) conditioned on the edges of G in F_t, but *not* conditioning on the resampled edges of H. We get

$$\begin{aligned} 1 &\geq \mathrm{E}\left[\, |\chi(G)-\chi(H)| \;\mid\; F_t \,\right] \\ &\geq |\, \mathrm{E}[\,\chi(G) \mid F_t\,] - \mathrm{E}[\,\chi(H) \mid F_t\,] \,|, \end{aligned}$$

by Jensen's inequality (Fact B.4.2). Now the key observation is that $\mathrm{E}[\,\chi(H) \mid F_t\,] = \mathrm{E}[\,\chi(G) \mid F_{t-1}\,]$. This is because H has the same distribution as G, and we have *not* conditioned on H's edges in E_t. So

$$\begin{aligned} 1 &\geq |\mathrm{E}[\,\chi(G) \mid F_t\,] - \mathrm{E}[\,\chi(G) \mid F_{t-1}\,]| \\ &= |S_t - S_{t-1}|. \end{aligned}$$ □

Claim 29.4.4 shows that the hypotheses of Azuma's inequality (Theorem 29.3.1) are satisfied. We obtain that

$$\Pr\left[\, |\chi(G) - \mathrm{E}[\,\chi(G)\,]| \geq c\sqrt{n} \,\right] \;=\; \Pr\left[\, |S_n - S_0| \geq c\sqrt{n} \,\right] \;\leq\; 2\exp(-c^2/2).$$

This proves Theorem 29.4.3.

29.5 Proof of Azuma's inequality

Let $\lambda > 0$ be an arbitrary parameter. A useful viewpoint is to define the following random variables.

$$X_t \;=\; \exp\left(\lambda(S_t - S_0) - \lambda^2 t/2\right) \qquad \forall t \in \{0,\ldots,n\}$$

This might seem a strange definition, but there are some fundamental reasons why it is very natural. In more advanced contexts, it is called the stochastic exponential.

The key is the following claim, after which the remainder of the proof is straightforward.

Claim 29.5.1. The RVs $X_0, X_1, \ldots, X_n$ are a *super*martingale.

Now we follow Chernoff's approach of Section 21.2.1.

$$\begin{aligned} \Pr[\, S_n - S_0 \geq \alpha \,] &= \Pr\left[\exp\left(\lambda(S_n-S_0) - \lambda^2 n/2\right) \geq \exp(\lambda\alpha - \lambda^2 n/2)\,\right] && \text{(by monotonicity)} \\ &\leq \frac{\mathrm{E}[\,X_n\,]}{\exp(\lambda\alpha - \lambda^2 n/2)} && \text{(by Markov's inequality)} \\ &\leq \frac{\mathrm{E}[\,X_0\,]}{\exp(\lambda\alpha - \lambda^2 n/2)} && \text{(by Claims 29.5.1 and 29.2.10)} \\ &= \exp(-\lambda\alpha + \lambda^2 n/2) && \text{(since } X_0 = 1\text{)} \\ &= \exp(-\alpha^2/n + \alpha^2/2n) \;=\; \exp(-\alpha^2/2n) && \text{(by plugging in } \lambda = \alpha/n\text{).} \end{aligned}$$

This completes the proof of Theorem 29.3.1. It remains to prove the claim.

Proof of Claim 29.5.1. Consider any time t. Then, by definition of X_t,

$$\mathrm{E}\left[\, X_t \;\middle|\; F_{t-1} \,\right] \;=\; \mathrm{E}\left[\exp\big(\lambda(S_t - S_0) - \lambda^2 t/2\big) \;\middle|\; F_{t-1}\right]$$

Observe that the random variables S_{t-1} and S_0 are completely determined by F_{t-1}, and so $e^{\lambda(S_{t-1}-S_0)-\lambda^2 t/2}$ is also. By Claim 29.2.5 we can pull that outside the expectation.

$$=\; e^{\lambda(S_{t-1}-S_0)-\lambda^2 t/2} \cdot \mathrm{E}\left[\exp\big(\lambda(S_t - S_{t-1})\big) \;\middle|\; F_{t-1}\right]$$

Since $|S_t - S_{t-1}| \leq 1$ and $\mathrm{E}\left[\, S_t - S_{t-1} \;\middle|\; F_{t-1} \,\right] = 0$, the hypotheses of Hoeffding's lemma (Lemma 21.3.2) are satisfied, so we obtain the upper bound

$$\leq\; e^{\lambda(S_{t-1}-S_0)-\lambda^2 t/2} \cdot e^{\lambda^2/2} \;=\; X_{t-1}.$$

This shows that $X_0, \ldots, X_n$ form a supermartingale. □

29.6 Exercises

Exercise 29.1. Prove Claim 29.2.9.

Exercise 29.2. Let $S_0, S_1, \ldots, S_n$ be a martingale, as in Section 29.2. Suppose that there are non-random values $c_1, \ldots, c_n$ such that $|S_t - S_{t-1}| \leq c_t$ for all $t \in [n]$. Define

$$X_t \;=\; \exp\left(\lambda(S_t - S_0) - \frac{\lambda^2}{2}\sum_{i=1}^{t} c_i^2\right) \qquad \forall t \in \{0, \ldots, n\}\,.$$

Prove that $X_0, \ldots, X_n$ form a supermartingale.

Chapter 30

Gradient Descent

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function and let $\mathcal{X} \subseteq \mathbb{R}^n$ be a convex set. We will assume that $\min_{x \in \mathcal{X}} f(x)$ is achieved by some point x^*. Our objective is to find a point $x \in \mathcal{X}$ for which $f(x) - f(x^*)$ is small.

TODO: Define subgradients.

Since f is convex, we have $\partial f(x) \neq \emptyset$ for all $x \in \mathbb{R}^n$. Throughout this chapter we will use the Euclidean norm, and we will assume that f is 1-Lipschitz. This condition can be equivalently stated in terms of subgradients (via Fact B.3.10), which implies

$$\|g\|_2 \leq 1 \qquad \forall x \in \mathcal{X},\ g \in \partial f(x). \tag{30.0.1}$$

30.1 Unconstrained gradient descent

The algorithm is shown in Algorithm 30.1.

Algorithm 30.1 Gradient descent for minimizing a convex, 1-Lipschitz function over $\mathbb{R}^n$.

```
1: procedure GradientDescent(vector $x_1 \in \mathbb{R}^n$, int $T$)
2:     Let $\eta = 1/\sqrt{T}$
3:     for $i \leftarrow 1, \ldots, T$ do
4:         $x_{i+1} \leftarrow x_i - \eta g_i$, where $g_i = \nabla f(x_i)$ if $f$ is differentiable, or
5:                                  $g_i$ is any subgradient in $\partial f(x_i)$ if $f$ is non-differentiable
6:     return $\sum_{i=1}^{T} x_i / T$
```

Remark 30.1.1. Observe that the only way that Algorithm 30.1 accesses the function f is in line 4. We assume that f is presented to the algorithm as a ***subgradient oracle***, which receives a point $x \in \mathbb{R}^n$, and returns any subgradient $g \in \partial f(x)$.

Theorem 30.1.2. Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is convex and 1-Lipschitz. Fix an optimal solution $x^* \in \operatorname{argmin}_x f(x)$ and a starting point $x_1 \in \mathbb{R}^n$. Define $\eta = \frac{1}{\sqrt{T}}$. Suppose that $\|x_1 - x^*\|_2 \leq 1$. Then

$$f\left(\frac{1}{T}\sum_{i=1}^{T} x_i\right) - f(x^*) \;\leq\; \frac{1}{\sqrt{T}}.$$

References: (Bubeck, 2015, Theorem 3.2), (Shalev-Shwartz and Ben-David, 2014, Corollary 14.2, §14.2.3).

Proof. We bound the error on the i^{th} iteration as follows:

$$\begin{aligned}
f(x_i) - f(x^*) &\leq \langle\, g_i,\, x_i - x^* \,\rangle && \text{(by the subgradient inequality (B.3.1))}\\
&= \frac{1}{\eta}\langle\, x_i - x_{i+1},\, x_i - x^* \,\rangle && \text{(by the gradient step in line 4)}\\
&= \frac{1}{2\eta}\Big(\, \|x_i - x_{i+1}\|_2^2 + \|x_i - x^*\|_2^2 - \|x_{i+1} - x^*\|_2^2 \,\Big) && \text{(by the cosine law (B.2.1)).}
\end{aligned}$$

To analyze the average error, sum the previous displayed equation over i. The last two terms telescope, yielding

$$\begin{aligned}
\sum_{i=1}^{T}\big(f(x_i) - f(x^*)\big) &\leq \frac{1}{2\eta}\Bigg(\Big(\sum_{i=1}^{T}\|x_i - x_{i+1}\|_2^2\Big) + \|x_1 - x^*\|_2^2 - \|x_{T+1} - x^*\|_2^2\Bigg)\\
&\leq \frac{1}{2\eta}\Big(\sum_{i=1}^{T}\|\eta g_i\|_2^2 + \|x_1 - x^*\|_2^2\Big) \quad \text{(by the gradient step in line 4)}\\
&\leq \frac{\eta T}{2} + \frac{1}{2\eta} \qquad \text{(by (30.0.1) and the assumption on } x_1\text{)}
\end{aligned}$$

Dividing by T and using Jensen's inequality (Fact B.4.2) and the definition of η gives

$$f\Bigg(\sum_{i=1}^{T}\frac{x_i}{T}\Bigg) - f(x^*) \;\leq\; \sum_{i=1}^{T}\frac{1}{T}\big(f(x_i) - f(x^*)\big) \;\leq\; \frac{\eta}{2} + \frac{1}{2T\eta} \;=\; \frac{1}{\sqrt{T}},$$

as required. □

Remark 30.1.3. The error guarantee of Theorem 30.1.2 is optimal for *any* algorithm that only accesses f using a subgradient oracle (Bubeck, 2015, Theorem 3.13).

General reduction from arbitrary scaling. The analysis present above assumes that the given function f is 1-Lipschitz. How shall we handle a function that is L-Lipschitz? It also has a certain "scale assumption" $\|x_1 - x^*\|_2 \leq 1$. How could we handle a general scale, say $\|x_1 - x^*\|_2 \leq R$? In Section 30.4 we will discuss a general reduction that can handle such scenarios.

Theorem 30.1.4. Suppose that we have a theorem giving a convergence rate guarantee $c(T)$ for gradient descent assuming f is 1-Lipschitz and assuming the "scale" $\|x_1 - x^*\|_2 \leq 1$. Suppose h is an L-Lipschitz function whose "scale" is bounded by R. Then there is a black-box reduction from h to f, showing that gradient descent on h achieves convergence rate $RL \cdot c(T)$.

30.2 Projected gradient descent

In this section we consider the problem $\min_{x\in\mathcal{X}} f(x)$ where $\mathcal{X}$ is a closed, convex set. Again, f is assumed to be convex and 1-Lipschitz.

The ordinary gradient descent algorithm does not ensure that the iterates remain in $\mathcal{X}$. In this section we modify the algorithm to project back onto $\mathcal{X}$. The algorithm now takes a gradient step from the iterate x_i to compute a new point y_{i+1}, then projects onto $\mathcal{X}$ to obtain the new iterate x_{i+1}.

The algorithm, shown in Algorithm 30.2, is a slight modification of Algorithm 30.1. The theorem is a slight modification of Theorem 30.1.2. The only changes are highlighted below.

Algorithm 30.2 Projected gradient descent for minimizing convex, 1-Lipschitz functions over a convex set.

```
1: procedure ProjectedGradientDescent(set $\mathcal{X} \subseteq \mathbb{R}^n$, vector $x_1 \in \mathcal{X}$, int $T$)
2:     Let $\eta = 1/\sqrt{T}$
3:     for $i \leftarrow 1, \ldots, T$ do
4:         $y_{i+1} \leftarrow x_i - \eta g_i$, where $g_i \in \partial f(x_i)$.
5:         $x_{i+1} \leftarrow \Pi_{\mathcal{X}}(y_{i+1})$
6:     return $\sum_{i=1}^{T} x_i/T$
```

Theorem 30.2.1. Let $\mathcal{X} \subseteq \mathbb{R}^n$ be a convex set. Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is convex and 1-Lipschitz (with respect to $\|\cdot\|_2$). Fix an optimal solution $x^* \in \operatorname{argmin}_{x \in \mathcal{X}} f(x)$ and a starting point $x_1 \in \mathcal{X}$. Define $\eta = \frac{1}{\sqrt{T}}$. Suppose that $\|x_1 - x^*\|_2 \leq 1$. Then

$$f\left(\frac{1}{T}\sum_{i=1}^{T} x_i\right) - f(x^*) \;\leq\; \frac{1}{\sqrt{T}}.$$

References: (Bubeck, 2015, Theorem 3.2), (Shalev-Shwartz and Ben-David, 2014, §14.4.1).

Proof. We bound the error on the i^{th} iteration as follows:

$$\begin{aligned}
f(x_i) - f(x^*) &\leq \langle\, g_i,\, x_i - x^* \,\rangle && \text{(by the subgradient inequality (B.3.1))} \\
&= \frac{1}{\eta}\langle\, x_i - y_{i+1},\, x_i - x^* \,\rangle && \text{(by the gradient step in line 4)} \qquad (30.2.1) \\
&= \frac{1}{2\eta}\Big(\|x_i - y_{i+1}\|_2^2 + \|x_i - x^*\|_2^2 - \|y_{i+1} - x^*\|_2^2 \Big) && \text{(by the cosine law (B.2.1))} \\
&\leq \frac{1}{2\eta}\Big(\|x_i - y_{i+1}\|_2^2 + \|x_i - x^*\|_2^2 - \|x_{i+1} - x^*\|_2^2 \Big).
\end{aligned}$$

The last line uses Fact B.3.12: since x_{i+1} is the projected point $\Pi_{\mathcal{X}}(y_{i+1})$ and $x^* \in \mathcal{X}$, we have $\|x_{i+1} - x^*\|_2^2 \leq \|y_{i+1} - x^*\|_2^2$.

To analyze the average error, sum the previous displayed equation over i. The last two terms telescope, yielding

$$\begin{aligned}
\sum_{i=1}^{T} \big(f(x_i) - f(x^*)\big) &\leq \frac{1}{2\eta}\left(\left(\sum_{i=1}^{T} \|x_i - y_{i+1}\|_2^2 \right) + \|x_1 - x^*\|_2^2 - \|x_{T+1} - x^*\|_2^2 \right) \\
&\leq \frac{1}{2\eta}\left(\sum_{i=1}^{T} \|\eta g_i\|_2^2 + \|x_1 - x^*\|_2^2 \right) && \text{(by the gradient step in line 4)} \\
&\leq \frac{\eta T}{2} + \frac{1}{2\eta} && \text{(by (30.0.1) and the assumption on } x_1\text{)}
\end{aligned}$$

Dividing by T and using Jensen's inequality (Fact B.4.2) and the definition of η gives

$$f\left(\sum_{i=1}^{T} \frac{x_i}{T}\right) - f(x^*) \;\leq\; \sum_{i=1}^{T} \frac{1}{T}\big(f(x_i) - f(x^*)\big) \;\leq\; \frac{\eta}{2} + \frac{1}{2T\eta} \;=\; \frac{1}{\sqrt{T}},$$

as required. □

30.3 Stochastic gradient descent

Stochastic gradient descent is a generalization of gradient descent in which we relax the notion of a subgradient oracle. Instead of requiring that the oracle return an *actual* subgradient, we will allow it to return a random vector that *in expectation* is a subgradient. (This is formalized below.) The stochastic gradient descent algorithm, shown in Algorithm 30.3, is a trivial modification of Algorithm 30.2 to use this stochastic oracle.

Algorithm 30.3 Stochastic gradient descent for minimizing a convex, Lipschitz function over a convex set $\mathcal{X}$.

```
1: procedure StochasticGradientDescent(set $\mathcal{X} \subseteq \mathbb{R}^n$, vector $x_1 \in \mathcal{X}$, int $T$)
2:     Let $\eta = 1/\sqrt{T}$
3:     for $i \leftarrow 1, \ldots, T$ do
4:         Let $\hat{g}_i$ be a random vector obtained from the subgradient oracle at $x_i$
5:         $y_{i+1} \leftarrow x_i - \eta \hat{g}_i$,
6:         $x_{i+1} \leftarrow \Pi_{\mathcal{X}}(y_{i+1})$
7:     return $\sum_{i=1}^{T} x_i/T$
```

The analysis of stochastic gradient descent is actually quite straightforward. Below we will prove Theorem 30.3.2, which is very similar to Theorem 30.2.1, it just requires some care with conditional expectations.

With that in mind, let us use the notation of Section 29.2, in which F_t consists of all RVs created up to the end of iteration t of the algorithm. Then the full history is as follows.

$$\begin{aligned}
F_0 &= \emptyset \\
F_1 &= (\hat{g}_1) \\
F_2 &= (\hat{g}_1, \hat{g}_2) \\
F_3 &= (\hat{g}_1, \hat{g}_2, \hat{g}_3) \\
\vdots &= \vdots \\
F_t &= (\, \hat{g}_i \,:\, i \leq t \,) \qquad \forall t \in [T].
\end{aligned} \tag{30.3.1}$$

Claim 30.3.1. For each $i \in [T]$,

- $\hat{g}_i$ is completely determined by F_i, and
- x_{i+1} is completely determined by F_i.

Proof sketch. The first statement is trivial from (30.3.1). By induction, x_i is determined by F_{i-1}. Thus, x_i and $\hat{g}_i$ are both determined by F_i, from which it follows that y_{i+1} and x_{i+1} are too. □

Randomized subgradient oracle: assumptions. To analyze the algorithm, we require two assumptions about the subgradient oracle. For notational convenience, we define

$$g_i = \mathrm{E}[\, \hat{g}_i \mid F_{i-1} \,].$$

Our first assumption is that

$$\underbrace{\mathrm{E}[\, \hat{g}_i \mid F_{i-1} \,]}_{=g_i} \in \partial f(x_i). \tag{30.3.2}$$

That is, the *expected* output of the oracle is a subgradient.

Note that both sides of (30.3.2) are completely determined by F_{i-1}. For the left, this follows from definition of conditional expectation (see Definition 29.2.2). For the right, this follows from Claim 30.3.1.

Our second assumption is that the vectors returned by the subgradient oracle are not too large. We formalize this assumption as:

$$\mathrm{E}\left[\, \|\hat{g}_i\|_2^2 \,\right] \;\leq\; 1 \qquad \forall i. \tag{30.3.3}$$

Theorem 30.3.2. Let $\mathcal{X} \subseteq \mathbb{R}^n$ be a convex set. Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is convex. Fix an optimal solution $x^* \in \operatorname{argmin}_{x \in \mathcal{X}} f(x)$ and a starting point $x_1 \in \mathcal{X}$. Define $\eta = \frac{1}{\sqrt{T}}$. Suppose that $\|x_1 - x^*\|_2 \leq 1$. Under our assumptions on the subgradient oracle,

$$\mathrm{E}\left[f\left(\frac{1}{T} \sum_{i=1}^{T} x_i \right) \right] - f(x^*) \;\leq\; \frac{1}{\sqrt{T}}.$$

References: (Shalev-Shwartz and Ben-David, 2014, Theorem 14.8), (Bubeck, 2015, Section 6.1).

Before proving the theorem, it is useful to define the *noise*.

$$\text{Noise in subgradient oracle:} \qquad \hat{z}_i \;=\; g_i - \hat{g}_i.$$

Lemma 30.3.3.

$$\mathrm{E}\left[f\Big(\frac{1}{T} \sum_{i=1}^{T} x_i \Big) \right] - f(x^*) \;\leq\; \frac{1}{\sqrt{T}} + \frac{1}{T}\, \mathrm{E}\left[\sum_{i=1}^{T} \langle\, \hat{z}_i,\, x_i - x^* \,\rangle \right].$$

Proof. We bound the error on the i^{th} iteration as follows:

$$\begin{aligned} f(x_i) - f(x^*) \;&\leq\; \langle\, g_i,\, x_i - x^* \,\rangle && \text{(by the subgradient inequality (B.3.1))} \\ &\leq\; \langle\, \hat{g}_i,\, x_i - x^* \,\rangle + \langle\, \hat{z}_i,\, x_i - x^* \,\rangle && \text{(by definition of } \hat{z}_i) \\ &=\; \frac{1}{\eta} \langle\, x_i - y_{i+1},\, x_i - x^* \,\rangle + \langle\, \hat{z}_i,\, x_i - x^* \,\rangle && \text{(by the gradient step)} \end{aligned}$$

The first term is identical to the quantity considered in the previous section, equation (30.2.1). Thus, summing over all T and telescoping, we obtain

$$\sum_{i=1}^{T} \big(f(x_i) - f(x^*) \big) \;\leq\; \sum_{i=1}^{T} \frac{\|x_i - y_{i+1}\|_2^2}{2\eta} + \frac{\|x_1 - x^*\|_2^2}{2\eta} + \sum_{i=1}^{T} \langle\, \hat{z}_i,\, x_i - x^* \,\rangle$$

Now using that $x_i - y_{i+1} = \eta \hat{g}_i$ and $\|x_1 - x^*\| \leq 1$, we obtain

$$\leq\; \frac{\eta}{2} \sum_{i=1}^{T} \|\hat{g}_i\|_2^2 + \frac{1}{2\eta} + \sum_{i=1}^{T} \langle\, \hat{z}_i,\, x_i - x^* \,\rangle. \tag{30.3.4}$$

Now dividing by T and taking the expectation, we have

$$\mathrm{E}\left[\frac{1}{T} \sum_{i=1}^{T} \big(f(x_i) - f(x^*) \big) \right] \;\leq\; \frac{\eta}{2T} \sum_{i=1}^{T} \mathrm{E}\left[\, \|\hat{g}_i\|_2^2 \,\right] + \frac{1}{2\eta T} + \mathrm{E}\left[\frac{1}{T} \sum_{i=1}^{T} \langle\, \hat{z}_i,\, x_i - x^* \,\rangle \right].$$

Using our assumption (30.3.3) on the oracle, and Jensen's inequality (Fact B.4.2), we obtain

$$\mathrm{E}\left[f\Big(\sum_{i=1}^{T} \frac{x_i}{T} \Big) \right] - f(x^*) \;\leq\; \frac{\eta}{2} + \frac{1}{2\eta T} + \mathrm{E}\left[\frac{1}{T} \sum_{i=1}^{T} \langle\, \hat{z}_i,\, x_i - x^* \,\rangle \right].$$

Plugging in $\eta = 1/\sqrt{T}$ completes the proof. □

Claim 30.3.4. Define $S_0 = 0$ and

$$S_i \;=\; \sum_{j \leq i} \langle\, \hat{z}_j,\, x_j - x^* \,\rangle \qquad \forall i \in [T].$$

Then $S_0, S_1, \ldots, S_T$ is a martingale (with respect to the full history $F_0, F_1, \ldots, F_T$).

The proof is left as Exercise 30.1.

Proof of Theorem 30.3.2. Since $S_0, S_1, \ldots, S_T$ is a martingale, it follows from Corollary 29.2.8 that $\mathrm{E}[\, S_T \,] = \mathrm{E}[\, S_0 \,] = 0$. So by Lemma 30.3.3, we have

$$\mathrm{E}\left[f\Big(\frac{1}{T} \sum_{i=1}^{T} x_i \Big) \right] - f(x^*) \;\leq\; \frac{1}{\sqrt{T}} + \frac{1}{T}\, \mathrm{E}\Bigg[\underbrace{\sum_{i=1}^{T} \langle\, \hat{z}_i,\, x_i - x^* \,\rangle}_{=S_T} \Bigg] \;=\; \frac{1}{\sqrt{T}}.$$ □

30.3.1 Application: training ML models

To train a machine learning model, we often need to minimize a function $f : \mathbb{R}^n \to \mathbb{R}$ of the form

$$f(x) \;=\; \frac{1}{m} \sum_{i=1}^{m} f_i(x)$$

where each f_i is convex.

This setting commonly arises if $x \in \mathbb{R}^n$ gives some parameters of the model, there are m training examples, and f_i is a "loss function" that measures how well these parameters do at classifying the i^{th} example.

A prototypical example would be training a "soft" support vector machine, with the training set

Examples:	$z_1, \ldots, z_m \in \mathbb{R}^n$
Labels:	$y_1, \ldots, y_m \in \{+1, -1\}$

and loss function $f_i(x) = \max\{0, 1 - y_i \langle\, x,\, z_i \,\rangle\}$. If $\langle\, x,\, z_i \,\rangle$ has the opposite sign of y_i then $f_i(x)$ gives a positive loss; otherwise, if they have the same sign and $|\langle\, x,\, z_i \,\rangle| \geq 1$, then $f_i(x)$ gives no loss. Thus minimizing f yields parameters x for which the sign of $\langle\, x,\, z_i \,\rangle$ tends to match the label y_i.

References: (Shalev-Shwartz and Ben-David, 2014, Equation (15.6)).

Suppose we have a (deterministic) subgradient oracle for each f_i. We can build upon these to give a randomized subgradient oracle for f, with which we can execute SGD.

The randomized subgradient oracle. The randomized subgradient oracle for f is very simple. Given a fixed point x, pick an index $I \in [m]$ uniformly at random, then return any subgradient of f_I at the point x.

Question 30.3.5. Suppose that each function f_i is 1-Lipschitz. Is assumption (30.3.3) necessarily satisfied?

Answer.

Yes. Each vector g_i has $\|g_i\|_2 \leq 1$ by Fact B.3.10, which implies that $\|\hat{g}\|_2 \leq 1$ with probability 1. So $\|\hat{g}\|_2^2 \leq 1$, which implies (30.3.3).

Claim 30.3.6. $\mathrm{E}\left[\,\hat{g}\,\right] \in \partial f(x)$.

Proof. Let $s_i \in \partial f_i(x)$ be the vector that would be returned by the i^{th} subgradient oracle, so that $\mathrm{E}\left[\,\hat{g}\,\right] = \frac{1}{m}\sum_{i=1}^{m} s_i$. Then

$$\begin{aligned}
\mathrm{E}\left[\,\hat{g}\,\right] &\in \frac{1}{m}\sum_{i=1}^{m} \partial f_i(x) && \text{(definition of Minkowski sum)}\\
&= \partial\Big(\frac{1}{m}\sum_{i=1}^{m} f_i(x)\Big) && \text{(by Fact B.3.11)}\\
&= \partial\big(f(x)\big). \quad \square
\end{aligned}$$

30.3.2 Application: geometric median

Let $p_1, \ldots, p_m$ be points in $\mathbb{R}^n$. A geometric median of these points is a point x^* that minimizes

$$f(x) \;=\; \frac{1}{m}\sum_{i=1}^{m} \|x - p_i\|\,.$$

We present an application of SGD to approximate the geometric median. Following the notation of the previous section, let $f_i(x) = \|x - p_i\|$.

Claim 30.3.7. f_i is 1-Lipschitz.

Proof sketch. This follows from the triangle inequality. □

Claim 30.3.8. Define

$$g \;=\; \begin{cases} \frac{x-p_i}{\|x-p_i\|} & (\text{if } x \neq p_i)\\ 0 & (\text{if } x = p_i). \end{cases}$$

Then $g \in \partial f_i(x)$.

Proof sketch. If $x \neq p_i$ then f_i is differentiable at x, and one may verify that $g = \nabla f_i(x)$. Otherwise $x = p_i$, so x is the global minimizer of f_i, in which case $0 \in \partial f_i(x)$. □

Imagine that we have a *warm start* for finding the geometric median. Specifically, we have a point $x_1 \in \mathbb{R}^n$ satisfying

$$\|x_1 - x^*\| \leq \lambda \qquad \text{where} \qquad \lambda = 40 f(x^*).$$

Interestingly, there is a randomized algorithm to find this warm start in $O(n)$ time. See Appendix C.1 of this paper. Given the warm start, SGD provides the desired approximation.

Theorem 30.3.9. Given the warm start x_1, run stochastic gradient descent with no constraints (i.e., $\mathcal{X} = \mathbb{R}^n$) for $T = 1600/\epsilon^2$ iterations. Then

$$\mathrm{E}\left[\, f(x_T) \,\right] \;\leq\; (1+\epsilon)\cdot f(x^*).$$

Proof. Theorem 30.3.2 yields

$$\mathrm{E}\left[\, f(x_T) \,\right] - f(x^*) \;\leq\; \frac{\lambda}{\sqrt{T}} \;=\; \frac{40 f(x^*)}{\sqrt{T}} \;=\; \frac{40 f(x^*)}{\sqrt{1600/\epsilon^2}} \;=\; \epsilon f(x^*). \qquad \square$$

Question 30.3.10. What is the runtime of this algorithm?

Answer.

Each iteration of SGD requires $O(n)$ time, so the total runtime is $O(nT) = O(n/\epsilon^2)$.

30.4 Scaling reductions

Our analyses above make two assumptions

- *Scale of codomain:* the given function f is 1-Lipschitz, and
- *Scale of domain:* $\|x_1 - x^*\|_2 \leq 1$.

How can we handle a general scale, say an L-Lipschitz function with $\|x_1 - x^*\|_2 \leq R$? There is a general reduction that can handle such scenarios.

Meta-theorem. Suppose that we have a theorem giving a convergence rate guarantee $c(T)$ for gradient descent assuming that $f : \hat{\mathcal{X}} \to \mathbb{R}$ is 1-Lipschitz and assuming $\|x_1 - x^*\|_2 \leq 1$. Suppose that $h : \mathcal{X} \to \mathbb{R}$ is a convex function that is L-Lipschitz, and such that $\|x_1 - x^*\| \leq R$. Then there is a black-box reduction from h to f, showing that gradient descent on h achieves convergence rate $RL \cdot c(T)$.

Proof of meta-theorem. Let $OPT = \min_{x \in \mathcal{X}} h(x)$. Define $\hat{\mathcal{X}} = \mathcal{X}/R$ and $f : \hat{\mathcal{X}} \to \mathbb{R}$ by

$$f(x) \;=\; \frac{1}{RL}(h(Rx) - OPT).$$

Thus,

$$\begin{aligned} h(x) \;&=\; RL \cdot f(x/R) + OPT \qquad (30.4.1)\\ \min_{x \in \hat{\mathcal{X}}} f(x) \;&=\; 0. \end{aligned}$$

Claim 30.4.1. $v \in \partial h(x)$ iff $v/L \in \partial f(x/R)$.

Consider running gradient descent on h with step sizes $\eta_t = \frac{R}{L\sqrt{t}}$ from the starting point x_1, producing iterates $x_2, x_3, \ldots$. Let g_i be the subgradient used in the ith iteration. Define $\hat{g}_i = g_i/L$.

Simulataneously, imagine running gradient descent on f with step sizes $\hat{\eta}_t = \frac{1}{\sqrt{t}} = \frac{L}{R}\eta_t$ and vectors $\hat{g}_i = g_i/L$, from the starting point $\hat{x_1} = x_1/R$. Let $\hat{x_2}, \hat{x_3}, \ldots$ be the vectors produced.

Claim 30.4.2. $\hat{x_i} = x_i/R$ for all $i \geq 1$.

Proof. By induction, the case $i = 1$ true by definition. So suppose true up to i. By definition $g_i \in \partial h(x_i)$, so Claim 30.4.1 implies that $\hat{g}_i \in \partial f(x_i/R) = \partial f(\hat{x}_i)$. Then

$$\widehat{x_{i+1}} \;=\; \hat{x}_i - \hat{\eta}_i \cdot \hat{g}_i \;=\; \frac{1}{R}x_i - \frac{L}{R}\eta_i \cdot \frac{1}{L}g_i \;=\; \frac{1}{R}(x_i - \eta_i g_i) \;=\; \frac{1}{R}x_{i+1}. \qquad \square$$

To illustrate the meta-theorem, we apply it to Theorem 30.1.2, obtaining:

Theorem 30.4.3. Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is convex and L-Lipschitz (with respect to $\|\cdot\|_2$). Fix an optimal solution $x^* \in \operatorname{argmin}_x f(x)$ and a starting point $x_1 \in \mathbb{R}^n$. Define $\eta = \frac{L}{R\sqrt{T}}$. Suppose that $\|x_1 - x^*\|_2 \leq R$. Then

$$\begin{aligned} h\left(\sum_{i=1}^{T} \frac{x_i}{T}\right) - h(x^*) &= RL \cdot f\left(\sum_{i=1}^{T} \frac{x_i}{RT}\right) && \text{(by (30.4.1))} \\ &= RL \cdot f\left(\sum_{i=1}^{T} \frac{\hat{x}_i}{T}\right) && \text{(by Claim 30.4.2)} \\ &\leq \frac{RL}{\sqrt{T}} && \text{(by Theorem 30.1.2).} \end{aligned}$$

30.5 Exercises

Exercise 30.1 The martingale in SGD.

Part I. Prove that $\mathrm{E}[\, \hat{z}_i \mid F_{i-1} \,] = 0$.

Part II. Prove Claim 30.3.4.

Exercise 30.2 High-probability bound for SGD. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex, 1/2-Lipschitz function. Let $\mathcal{X} \subseteq \mathbb{R}^n$ be a convex set with $\operatorname{diam}(\mathcal{X}) = 1$, and assume $x_1 \in \mathcal{X}$.

Suppose that the subgradient oracle satisfies (30.3.2). However, instead assuming (30.3.3), we assume that the noise satisfies $\|\hat{z}_i\| \leq 1/2$.

Part I. Prove that $\|\hat{g}_i\|_2^2 \leq 1$.

Part II. Prove that

$$\Pr\left[\; f\Big(\frac{1}{T}\sum_{i=1}^{T} x_i\Big) - f(x^*) \;\geq\; \frac{1 + \sqrt{2\ln(1/\delta)}}{\sqrt{T}} \;\right] \;\leq\; \delta.$$

Chapter 31

The Lovász Local Lemma

The Lovász Local Lemma (LLL) is an intriguing method for analyzing events that are not independent, but have some restricted sort of dependencies. It is not as widely applicable as many of the other the techniques we have seen so far, but from time to time one encounters scenarios in which the LLL is the only technique that works.

31.1 Statement of the Symmetric LLL

Very often when designing randomized algorithms, we create a discrete probability space in which there are "bad events" $\mathcal{E}_1, \ldots, \mathcal{E}_n$ that we do not want to occur. For example, in the congestion minimization problem (Section 22.2), $\mathcal{E}_i$ could be the event that edge i has too much congestion. Often our analysis aims to show that we can avoid these bad events with high probability, i.e., $\Pr\left[\bigwedge_{i=1}^n \overline{\mathcal{E}_i}\right] \approx 1$.

In this chapter we consider the weaker goal of showing that $\Pr\left[\bigwedge_{i=1}^n \overline{\mathcal{E}_i}\right] > 0$. There are two cases in which this goal is particularly simple.

- **Mutually independent events.** Suppose that the events are mutually independent, and that $\Pr[\mathcal{E}_i] < 1$ for every i. Then

$$\Pr\left[\bigwedge_{i=1}^n \overline{\mathcal{E}_i}\right] \;=\; \prod_{i=1}^{n} \Pr\left[\overline{\mathcal{E}_i}\right] \;>\; 0. \tag{31.1.1}$$

  For example, suppose we pick n digits at random and let $\mathcal{E}_i$ = "i^{th} digit is non-zero". Then $\bigwedge_{i=1}^n \overline{\mathcal{E}_i}$ is the event that all digits are zeros. This event does indeed happen with positive probability, due to (31.1.1).

- **Union bound works.** Suppose that $\sum_{i=1}^n \Pr[\mathcal{E}_i] < 1$. Then, by a union bound (see (A.3.1)),

$$\Pr\left[\bigwedge_{i=1}^n \overline{\mathcal{E}_i}\right] \;=\; 1 - \Pr\left[\bigvee_{i=1}^n \mathcal{E}_i\right] \;\geq\; 1 - \sum_{i=1}^{n} \Pr[\mathcal{E}_i] \;>\; 0.$$

If neither of these scenarios applies, then there are not many general-purpose techniques to try. The Lovász Local Lemma (LLL) is one of the few, and it has intriguing applications for a range of problems.

Roughly speaking, the LLL is applicable in scenarios where the $\mathcal{E}_i$'s are not mutually independent, but they can have some sort of limited dependencies. Formally, a ***dependency graph*** for events $\mathcal{E}_1, \ldots, \mathcal{E}_n$

is defined as follows. The vertex set is $[n]$. The neighbors of vertex i (excluding i itself) are denoted $\Gamma(i)$, and we also define $\Gamma^+(i) = \Gamma(i) \cup \{i\}$. The key requirement is that event $\mathcal{E}_i$ is independent from $\{\, \mathcal{E}_j \,:\, j \notin \Gamma^+(i) \,\}$, the events that are not neighbors of i. This last condition is stated more precisely as

$$\Pr[\,\mathcal{E}_i\,] \;=\; \Pr\left[\mathcal{E}_i \;\middle|\; \bigcap_{j \in J} \mathcal{E}_j\right] \qquad \text{for all } J \subseteq [n] \setminus \Gamma^+(i).$$

So, regardless of whether some of the events outside $\Gamma^+(i)$ occur, the probability of $\mathcal{E}_i$ occurring is unaffected.

Theorem 31.1.1 (The Symmetric LLL). Suppose that there is a dependency graph of maximum degree d. If $\Pr[\,\mathcal{E}_i\,] \leq p$ for every i and

$$p \cdot e \cdot (d+1) \;\leq\; 1 \tag{SLL}$$

then $\Pr\left[\bigcap_{i=1}^n \overline{\mathcal{E}_i}\right] \geq \left(\frac{d}{d+1}\right)^n > 0$.

References: (Motwani and Raghavan, 1995, Corollary 5.12), (Alon and Spencer, 2000, Corollary 5.1.2), (Mitzenmacher and Upfal, 2005, Theorem 6.11), Wikipedia.

Remark 31.1.2.

- Since $\left(\frac{d}{d+1}\right)^n \approx \exp(-n/d)$, due to Fact A.2.5, the LLL only gives an exponentially small probability of avoiding $\mathcal{E}_1, \ldots, \mathcal{E}_n$.
- The condition (SLL) can be improved to $ped \leq 1$. See, e.g., Harvey & Vondrák, Lemma 9.

31.2 Application: k-SAT

Instead, we will illustrate the LLL by considering a concrete application of it in showing satisfiability of k-CNF Boolean formulas. Recall that a k-CNF formula is a Boolean formula, involving any finite number of variables, where the formula is a *conjunction* ("and") of any number of clauses, each of which is a *disjunction* ("or") of exactly k *literals* (a variable or its negation). Let us assume that, for each clause, the variables appearing in it are all distinct.

For example, here is a 3-CNF formula with three variables and eight clauses.

$$\begin{aligned}\phi(a,b,c) \;&=\; (a \cup b \cup c) \cap (a \cup b \cup \overline{c}) \cap (a \cup \overline{b} \cup c) \cap (a \cup \overline{b} \cup \overline{c}) \cap \\ &\quad\; (\overline{a} \cup b \cup c) \cap (\overline{a} \cup b \cup \overline{c}) \cap (\overline{a} \cup \overline{b} \cup c) \cap (\overline{a} \cup \overline{b} \cup \overline{c})\end{aligned}$$

This formula is obviously unsatisfiable. One can easily generalize this construction to get an unsatisfiable k-CNF formula with k variables and 2^k clauses. Our next theorem says: the reason this formula is unsatisfiable is that we allowed each variable to appear in too many clauses.

Theorem 31.2.1. Let ϕ be a k-CNF formula where each variable appears in at most $2^k/ek$ clauses. Then ϕ is satisfiable.

Proof. Consider the probability space in which each variable is independently set to true or false with equal probability. A clause is not satisfied if every literal appearing in that clause is false. (For example, $\overline{a} \cup b \cup \overline{c}$ is unsatisfied if a is true, b is false, and c is true.) This happens with probability 2^{-k}, since the clause involves k distinct variables.

Let $\mathcal{E}_i$ be the event that the ith clause is unsatisfied. We have just argued that

$$\Pr[\,\mathcal{E}_i\,] \;=\; 2^{-k} \;=:\; p.$$

Consider the graph defined on $[n]$ in which there is an edge $\{i,j\}$ if some variable appears in both clause i and clause j. It is easy to see that this is a dependency graph: whether clause i is satisfied is independent from the clauses sharing no variables with clause i.

Each variable in clause i appears in at most $2^k/ek-1$ other clauses. The number of neighbors of clause i is at most k times larger, since it contains k variables. That is,

$$|\Gamma(i)| \;\leq\; 2^k/e-1 \;=:\; d.$$

Since $pe(d+1) \leq 1$, condition (SLL) is satisfied and $\Pr\left[\bigcap_{i=1}^n \overline{\mathcal{E}_i}\right] > 0$. So, if we pick values for the variables at random, there is positive probability of satisfying ϕ. This shows that ϕ is satisfiable. $\square$

31.3 Symmetric LLL: a proof sketch

In this section we give a sketchy proof of the symmetric LLL that tries to explain the sequence of ideas that leads to the proof. If this sketch is not to your taste, a correct and concise proof is given in Section 31.4.1. Instead of assuming (SLL), it will be convenient to assume the strengthened hypothesis

$$4pd \;\leq\; 1. \tag{31.3.1}$$

To make the notation more meaningful, let B_i denote the "bad" event $\mathcal{E}_i$ and let G_i be the "good" event $\overline{B_i}$. Note that independence from B_i is equivalent to independence from G_i. For any set $S \subseteq [n]$, let $G_S = \cap_{i\in S} G_i$.

The objective of the proof is to show that $\Pr\left[G_{[n]}\right] > 0$, i.e., with positive probability, all good events occur simultaneously. Note that the union bound gives the easy lower bound

$$\Pr[G_S] \;=\; 1-\Pr[\cup_{i\in S}B_i] \;\geq\; 1-\sum_{i\in S}\Pr[B_i] \;\geq\; 1-|S|p, \tag{31.3.2}$$

but in our scenario, this is too weak — we could have $|S|p \gg 1$. Instead, a natural idea is to use the chain rule (Fact A.3.5) to break apart this large conjunction:

$$\Pr\left[G_{[n]}\right] \;=\; \prod_{k=1}^{n}\Pr\left[G_k \mid G_{[k-1]}\right]. \tag{31.3.3}$$

We just need to show that each factor in this product is strictly positive.

Idea 1: Rather than showing that each factor is positive, it turns out to be convenient to negate the event and prove an upper bound. Specifically, we want to show that

$$\Pr[B_k \mid G_S] \;\leq\; \alpha p \qquad \forall S\subseteq[n] \tag{Hope}$$

for some α to be chosen later. In order for this conditional probability to be well defined, we need that $\Pr[G_S] > 0$. Let's ignore that for now as our whole purpose is to prove the stronger fact that $\Pr\left[G_{[n]}\right] > 0$. Trivially,

$$\Pr[B_k \mid G_S] \;=\; \Pr\left[B_k \mid G_{S\cap\Gamma(k)}\cap G_{S\setminus\Gamma(k)}\right].$$

Idea 2: We would like to somehow use the fact that B_k is independent of $G_{S\setminus\Gamma(k)}$ (due to the dependency graph). But it is difficult to use that fact due to the additional conditioning on $G_{S\cap\Gamma(k)}$. So as a first step we use the definition of conditional probability to write

$$\Pr[B_k \mid G_S] \;=\; \frac{\Pr\left[B_k\cap G_{S\cap\Gamma(k)} \mid G_{S\setminus\Gamma(k)}\right]}{\Pr\left[G_{S\cap\Gamma(k)} \mid G_{S\setminus\Gamma(k)}\right]}.$$

Idea 3: The next idea is to drop the "$\cap G_{S\cap\Gamma(k)}$" event yielding the following upper bound. We would hope that still this gives a good bound since $\Gamma(k)$ is small and $G_{S\cap\Gamma(k)}$ is a very likely event.

$$\Pr\left[B_k \mid G_S\right] \;\leq\; \frac{\Pr\left[B_k \mid G_{S\setminus\Gamma(k)}\right]}{\Pr\left[G_{S\cap\Gamma(k)} \mid G_{S\setminus\Gamma(k)}\right]} \;=\; \frac{\Pr\left[B_k\right]}{\Pr\left[G_{S\cap\Gamma(k)} \mid G_{S\setminus\Gamma(k)}\right]} \;\leq\; \frac{p}{\Pr\left[G_{S\cap\Gamma(k)} \mid G_{S\setminus\Gamma(k)}\right]}$$

The equality here uses our second idea, that B_k is independent of the events outside of $\Gamma(k)$.

Now, to prove (Hope), it suffices to prove that $\Pr\left[G_{S\cap\Gamma(k)} \mid G_{S\setminus\Gamma(k)}\right] \geq 1/\alpha$. The good news is that the conjunction $G_{S\cap\Gamma(k)}$ involves few events, so we are in good shape to use the union bound as in (31.3.2):

$$\Pr\left[G_{S\cap\Gamma(k)} \mid G_{S\setminus\Gamma(k)}\right] \;\geq\; 1 - \sum_{i\in S\cap\Gamma(k)} \Pr\left[B_i \mid G_{S\setminus\Gamma(k)}\right]. \tag{31.3.4}$$

If $S\cap\Gamma(k)=\emptyset$ then this quantity is 1 as the sum is empty. Otherwise, $|S\setminus\Gamma(k)|<|S|$, so we can use induction on $|S|$. We have

$$\begin{aligned}
\Pr\left[G_{S\cap\Gamma(k)} \mid G_{S\setminus\Gamma(k)}\right] \;&\geq\; 1 - \sum_{i\in\Gamma(k)} \Pr\left[B_i \mid G_{S\setminus\Gamma(k)}\right] && \text{(by (31.3.4))}\\
&\geq\; 1 - d\alpha p && \text{(inductively using (Hope), and } |\Gamma(k)|\leq d)\\
&\geq\; 1-\alpha/4 && \text{(by the strengthened hypothesis (31.3.1))}\\
&=\; 1/\alpha
\end{aligned}$$

if we now choose $\alpha=2$. This shows that (Hope) is satisfied.

31.4 The General LLL

The Symmetric Local Lemma is useful, but often somewhat restrictive. It works best when all events are equally probable, and when all neighborhood sizes are the same. We might want to consider scenarios in which some events are quite likely and some are quite rare. The likely events would need to depend on few other events, and the rare events could perhaps depend on many other events. There is a general form of the local lemma that can handle such scenarios.

Theorem 31.4.1 (General LLL). Suppose that there is a dependency graph and an $x\in(0,1)^n$ satisfying

$$\Pr\left[\mathcal{E}_i\right] \;\leq\; x_i\cdot\prod_{j\in\Gamma(i)}(1-x_j) \qquad \forall i. \tag{GLL}$$

Then $\Pr\left[\bigcap_{i=1}^n \overline{\mathcal{E}_i}\right] \geq \prod_{i=1}^n(1-x_i) > 0$.

References: (Motwani and Raghavan, 1995, Theorem 5.11), (Alon and Spencer, 2000, Lemma 5.1.1), Wikipedia.

This form of the local lemma is confusing at first because it's not obvious what these x_i values should be. In order to satisfy (GLL), on the right-hand side we want x_i to be big and each x_j to be small. Due to that tension, care is needed in finding the right x_i.

31.4.1 A Concise Proof of Theorem 31.4.1

We simultaneously prove by induction on $|S|$ that, for all $S\subseteq[n]$,

$$\Pr\left[G_S\right] > 0 \tag{31.4.1a}$$
$$\Pr\left[B_k \mid G_S\right] \leq x_k. \tag{31.4.1b}$$

In the case $S = \emptyset$, (31.4.1a) is trivial and (31.4.1b) follows directly from (GLL).

By relabeling, we may assume that $S = \{1, \ldots, s\}$, so

$$\Pr[G_S] = \prod_{j=1}^{s} \Pr\left[G_j \mid G_{\{1,\ldots,j-1\}}\right] = \prod_{j=1}^{s}\left(1 - \Pr\left[B_j \mid G_{\{1,\ldots,j-1\}}\right]\right) \geq \prod_{j=1}^{s}(1 - x_j),$$

where we inductively use (31.4.1a) to ensure that the conditional probabilities are well-defined, and we inductively use (31.4.1b) to provide the inequality. This proves (31.4.1a).

Next consider (31.4.1b). If $S \cap \Gamma(k) = \emptyset$ then $\Pr[B_k \mid G_S] = \Pr[B_k]$, so (31.4.1b) follows directly from (GLL). Otherwise, relabeling so that $S \cap \Gamma(k) = \{1, \ldots, t\}$, we have

$$\begin{aligned}
\Pr[B_k \mid G_S] &= \Pr\left[B_k \mid G_{S\cap\Gamma(k)} \cap G_{S\setminus\Gamma(k)}\right] = \frac{\Pr\left[B_k \cap G_{S\cap\Gamma(k)} \mid G_{S\setminus\Gamma(k)}\right]}{\Pr\left[G_{S\cap\Gamma(k)} \mid G_{S\setminus\Gamma(k)}\right]} \\
&\leq \frac{\Pr\left[B_k \mid G_{S\setminus\Gamma(k)}\right]}{\Pr\left[G_{S\cap\Gamma(k)} \mid G_{S\setminus\Gamma(k)}\right]} = \frac{\Pr[B_k]}{\prod_{j=1}^{t} \Pr\left[G_j \mid G_{(S\setminus\Gamma(k))\cup\{1,\ldots,j-1\}}\right]} \\
&\leq \frac{x_k \cdot \prod_{i\in\Gamma(k)}(1 - x_i)}{\prod_{j=1}^{t}\left(1 - \Pr\left[B_j \mid G_{(S\setminus\Gamma(k))\cup\{1,\ldots,j-1\}}\right]\right)} \leq \frac{x_k \cdot \prod_{i\in\Gamma(k)}(1 - x_i)}{\prod_{j=1}^{t}(1 - x_j)}.
\end{aligned}$$

The second inequality uses (GLL) and the third uses induction. The last expression is clearly at most x_k, proving (31.4.1b).

31.4.2 General implies Symmetric

Finally, let us conclude by noting that Theorem 31.4.1 implies Theorem 31.1.1. To see this, consider an instance satisfying (SLL). Set $x_i = 1/(d+1)$. The RHS of (GLL) is

$$\begin{aligned}
x_i \cdot \prod_{j\in\Gamma(i)}(1 - x_i) &\geq \frac{1}{d+1} \cdot \left(1 - \frac{1}{d+1}\right)^d \\
&\geq \frac{1}{e(d+1)} \qquad \text{(by calculus)} \\
&\geq p \qquad \text{(by (SLL))}.
\end{aligned}$$

This shows that (GLL) is satisfied, so Theorem 31.4.1 implies

$$\Pr\left[\bigcap_{i=1}^{n} \overline{\mathcal{E}_i}\right] \geq \prod_{i=1}^{n}(1 - x_i) = \left(\frac{d}{d+1}\right)^n,$$

which is the conclusion of Theorem 31.1.1.

31.5 An Algorithmic Local Lemma

As stated above, the LLL asserts the existence of a point in a probability space that simultaneously avoids certain bad events, assuming that their probabilities and dependencies are small enough. However, it does *not* suggest an efficient method to *find* such a point. This naturally leads to a research question:

is there an algorithmic form of the LLL?

This questioned had been studied for decades, culminating in a significant breakthrough by Robin Moser in 2009.

Suppose we wanted to actually find an point in the probability space avoiding the bad events. If we repeatedly picked independent samples from the underlying probability distribution, the expected time to find a point avoiding the bad events could be exponential in n. Usually we would like to find this point in time $\mathrm{poly}(n)$. This can be done for essentially all known applications of the local lemma.

Today we will discuss an algorithmic LLL which is not quite that general. Instead, we will focus on the application of the LLL to showing that SAT formulae are satisfiable, as discussed last time.

Theorem 31.5.1. There is a universal constant $\alpha \geq 1$ such that the following is true. Let ϕ be a k-CNF formula where each variable appears in at most $T := 2^{k-\alpha}/k$ clauses. Then ϕ is satisfiable. Moreover, there is a randomized, polynomial time algorithm to find a satisfying assignment.

The theorem is stronger when α is small. The proof that we will present can be optimized to get $\alpha = 3$. The existential result from last time achieves $\alpha = \lg_2(e) \approx 1.44$, which is essentially optimal. So today's result is weaker only by a small constant factor.

The algorithm proving the theorem is extremely simple, and algorithms of this sort were certainly considered decades ago. But, they were not analyzed until 2009, when the audacious and brilliant graduate student Robin Moser at ETH made a tremendous breakthrough. Related ideas were independently discovered by Pascal Schweitzer, also a graduate student.

References: Moser's PhD Thesis Section 2.5, Schweitzer's paper.

31.5.1 The Algorithm

Moser's algorithm is given in Algorithm 31.1.

Algorithm 31.1 Algorithm to find a satisfying assignment for a k-SAT formula ϕ, under the conditions of Theorem 31.5.1.

```
1: function SOLVE($\phi$)
2:     Randomly pick $\{0,1\}$ values for each variable in $\phi$        ▷ This uses $n$ random bits
3:     while there is an unsatisfied clause $C$
4:         FIX($C$)
5:     return the variable assignment
6: end function
7: function FIX($C$)
8:     Set each variable in $C$ to 0 or 1 randomly and independently     ▷ This uses $k$ random bits
9:     while there is an unsatisfied clause $D$ sharing some variable with $C$
10:        FIX($D$)                                                     ▷ Possibly $D = C$
11: end function
```

Notation. Let n be the number of variables and m be the number of clauses in ϕ. Let $T := 2^{k-\alpha}/k$ be the maximum number of occurrences of each variable. Each clause contains k variables, each of which can appear in only $T-1$ other clauses. So each clause shares a variable with less than $R := kT = 2^{k-\alpha}$ other clauses.

Claim 31.5.2. Consider any call to Fix that terminates. Let V be the set of variables that are assigned a different value after the call than before the call. Then every clause containing a variable in V is satisfied after the call.

Proof. Consider some clause D that contains a variable in V but is not satisfied after the call terminates. Consider the last time that any variable x in D was resampled. This must have happened during some call to Fix(E), for some clause E that also contains x. But Fix(E) would not terminate until D was satisfied, which is a contradiction. □

Corollary 31.5.3. Consider any call to Fix that terminates. Every clause that was satisfied *before* the call is still satisfied *after* the call completes.

Proof. If none of its variables changed, it is still satisfied. If at least one of its variables changed, Claim 31.5.2 applies. □

Corollary 31.5.4. Assume that C is violated, and consider any call to Fix(C) that terminates. The number of satisfied clauses before the call is strictly more than the number of satisfied clauses after the call.

Proof. By Corollary 31.5.3, the number of satisfied clauses cannot decrease. Furthermore, clause C must certainly be satisfied afterwards in order for Fix(C) to terminate. □

Corollary 31.5.5. Solve calls Fix at most m times. If the algorithm terminates, the output is a satisfying assignment.

Proof. By Corollary 31.5.4, every call from Solve to Fix(C) that terminates increases the number of satisfied clauses by at least one. The first claim follows since there are m clauses. The second claim is obvious: the only way that Solve can terminate is that all clauses are simultaneously satisfied. □

So it remains to show that, with high probability, every call to Fix terminates. The analysis of the algorithm is quite unusual, and counterintuitive the first time one sees it.

31.5.2 Incompressibility

From experience with software tools like `gzip`, one is likely familiar with the statement that "random information cannot be compressed". Let us now formalize that fact by stating that *some* strings can be compressed, but only a small fraction of them.

Claim 31.5.6. Let $x \in \{0,1\}^{\ell}$ be a uniformly random bit string of length ℓ. The probability that x can be compressed by $\log(1/\delta)$ bits is at most δ.

Proof. Consider any deterministic algorithm for encoding all bit strings of length ℓ into bit strings of arbitrary length. The number of bit strings that are encoded into $\ell - b$ bits is at most $2^{\ell-b}$. So, a random bit string has probability 2^{-b} of being encoded into $\ell - b$ bits. □

One can view this as a simple special case of the Kraft inequality.

31.5.3 Analysis of Algorithm 31.1

Theorem 31.5.7. Let $s = m \cdot (\log m + c) + \log(1/\delta)$ where c is a sufficiently large constant. Then the probability that the algorithm makes more than s calls to Fix (including both the top-level and recursive calls) is at most δ.

The proof proceeds by considering the interactions between two agents: the "CPU" and the "Debugger". The CPU runs the algorithm, periodically sending messages to the Debugger (we describe these messages in more detail below). However, if Fix gets called more than s times the CPU interrupts the execution and halts the algorithm.

The CPU needs n bits of randomness to generate the initial assignment in Solve, and needs k bits to regenerate variables in each call to Fix. Since the CPU will not execute Fix more than s times, it might as well generate all its random bits at the very start of the algorithm. So the first step performed by the CPU is to generate a random bitstring x of length $n + sk$ to provide all the randomness used in executing the algorithm.

The messages sent from the CPU to the Debugger are as follows.

- Every time the CPU runs Fix(C), it sends a message containing the identity of the clause C, and an extra bit indicating whether this is a top-level Fix (i.e., a call from Solve) or a recursive Fix.
- Every time Fix(C) finishes the CPU sends a message stating "recursive call finished".
- If Fix gets called s times, the CPU sends a message to the Debugger containing the current $\{0, 1\}$ assignment of all n variables.

Because the Debugger is notified when every call to Fix starts or finishes, it always knows which clause is currently being processed by Fix. A crucial detail is to figure out how many bits of communication are required to send these messages.

- For a top-level Fix, $\log m + O(1)$ bits suffice because there are only m clauses in ϕ.
- For a recursive Fix, $\log R + O(1)$ bits suffice because the Debugger already knows what clause is currently being fixed, and that clause shares variables with only R other clauses, so only R possible clauses could be passed to the next call to Fix.
- When each call to Fix(C) finishes, the corresponding message takes $O(1)$ bits.
- When Fix gets called s times, the corresponding message takes $n + O(1)$ bits.

The main point of the proof is to show that, if Fix gets called s times, then these messages reveal the random string x to the Debugger.

Since each clause is a *disjunction* (an "or" of k literals), there is *exactly one* assignment to those variables that does not satisfy the clause. So, whenever the CPU tells the Debugger that it is calling Fix(C), the Debugger knows exactly what the current assignment to C is. So, starting from the assignment that the Debugger received in the final message, it can work backwards and figure out what the previous assignment was before calling Fix. Repeating this process, it can figure out how the variables were set in each call to Fix, and also what the initial assignment was. Thus the Debugger can reconstruct the random string x.

The total number of bits sent by the CPU are

- $m(\log m + O(1))$ bits for all the messages sent when SOLVE calls FIX.
- $s \cdot (\log R + O(1))$ for all the messages sent in the $\leq s$ recursive calls.
- $n + O(1)$ bits to send the final assignment.

Let's ignore the $O(1)$ terms, which can be eliminated by choosing the constant c and α appropriately. Then x has been compressed from $n + sk$ bits to

$$m \log m \ + \ s \log R \ + \ n \quad \text{bits.}$$

This is an overall shrinking of

$$\begin{aligned} & \big(n + sk\big) \ - \ \big(m \log m + s \log R + n\big) \\ &= s(k - \log R) \ - \ m \log m \\ &= s\alpha \ - \ m \log m && \text{(since } R = 2^{k-\alpha}\text{)} \\ &= \big(m(\log m + c) + \log(1/\delta)\big)\alpha \ - \ m \log m && \text{(definition of } s\text{)} \\ &\geq \ \log(1/\delta) \end{aligned}$$

bits, assuming that c and α are sufficiently big constants.

We have argued that, if FIX gets called s times, then x can be compressed by $\log(1/\delta)$ bits. By Claim 31.5.6, this is possible with probability at most δ.

31.6 Exercises

Exercise 31.1 Local lemma for 0-1 matrices. Let M be a matrix with m rows, n columns such that

- every entry $M_{i,j} \in \{0, 1\}$,
- every row sums to r (i.e., $\sum_{j=1}^{n} M_{i,j} = r$ for all i),
- every column sums to c (i.e., $\sum_{i=1}^{m} M_{i,j} = c$ for all j.)

Show that there exists a vector $Y \in \{0, 1\}^n$ such that, letting $Z = M \cdot Y$, we have

$$\begin{aligned} \max_i Z_i \ &\leq \ (r/2) + O\big(\sqrt{r \log(rc)}\big) \\ \min_i Z_i \ &\geq \ (r/2) - O\big(\sqrt{r \log(rc)}\big). \end{aligned}$$

Part I

Back matter

Acknowledgements

Thanks to my teaching assistant (Victor) and various students over the years for many ideas, suggestions, corrections, and exercises.

Appendix B

Mathematical Background

B.1 Miscellaneous Facts

Fact B.1.1 (Stirling's Approximation).

$$e\left(\frac{n}{e}\right)^n \;<\; n! \;<\; en\left(\frac{n}{e}\right)^n$$

References: (Lehman et al., 2018, Theorem 14.5.1), Wikipedia.

Fact B.1.2 (Bounds on binomial coefficients). For any integers n, k with $1 \leq k \leq n$,

$$\left(\frac{n}{k}\right)^k \;\leq\; \binom{n}{k} \;\leq\; \sum_{i=0}^{k} \binom{n}{i} \;\leq\; \left(\frac{ne}{k}\right)^k.$$

References: (Cormen et al., 2001, page 1186), (Vershynin, 2018, Exercise 0.0.5), (Motwani and Raghavan, 1995, Proposition B.2), (Shalev-Shwartz and Ben-David, 2014, Lemma A.5), Wikipedia.

Fact B.1.3. For $0 \leq \epsilon \leq 1$, we have $(1+\epsilon)^2 \leq 1 + 3\epsilon$.

Proof. Expand $(1+\epsilon)^2$ as $1 + 2\epsilon + \epsilon^2$, and use $\epsilon^2 \leq \epsilon$ when $\epsilon \leq 1$. □

Claim B.1.4 (Separation principle). For any non-empty sets $Y_1, \ldots, Y_k$ and any functions $f_i : Y_i \to \mathbb{R}$, we have

$$\inf_{(y_1,\ldots,y_k)\in Y_1\times\cdots Y_k} \sum_{i\in[k]} f_i(y_i) \;=\; \sum_{i\in[k]} \inf_{y_i\in Y_i} f_i(y_i).$$

Proof. This follows from the property

$$\inf(A_1 + \cdots + A_k) \;=\; (\inf A_1) + \cdots + (\inf A_k), \tag{B.1.1}$$

applied to the sets $A_i = \{\, f_i(y_i) \,:\, y_i \in Y_i \,\}$. (On the left-hand side of (B.1.1), the "+" operation is Minkowski sum.) In turn, (B.1.1) follows from the elementary fact $\inf(A+B) = \inf(A) + \inf(B)$ by induction. See, e.g., equation (1.2) in Appendix A of (Hiriart-Urruty and Lemaréchal, 1996) or Wikipedia □

B.2 Geometry and norms

Norms. For a vector $x \in \mathbb{R}^d$, its ℓ_p norm is defined to be

$$\begin{aligned} \|x\|_p &= \Big(\sum_{i=1}^d |x_i|^p\Big)^{1/p} \qquad \forall p \in [1,\infty) \\ \|x\|_\infty &= \max_{1 \leq i \leq d} |x_i| \end{aligned}$$

Every norm $\|\cdot\|$ satisfies the

$$\text{Triangle inequality:} \qquad \|a+b\| \leq \|a\| + \|b\|.$$

The ℓ_2 norm is also called the ***Euclidean norm***. One useful identity it satisfies is

$$\|a-b\|_2^2 = \|a\|_2^2 - 2a^\mathsf{T} b + \|b\|_2^2 \qquad \forall a,b \in \mathbb{R}^n. \tag{B.2.1}$$

This identity might not have a canonical name, but some suggestions I have heard include the *generalized Pythagoras identity*, the *law of cosines*, or simply *completing the square*.

References: This is immediate from the bilinearity of an inner product. See, e.g., Apostol "Calculus, Volume II", page 17, or Wikipedia.

Let

$$B(p,r) = \{\, x : \|p-x\| \leq r \,\} \tag{B.2.2}$$

denote the Euclidean ball in $\mathbb{R}^d$ around p of radius r. If d is even, the ***volume*** of $B(p,r)$ is

$$\operatorname{vol} B(p,r) = \frac{\pi^{d/2} r^d}{(d/2)!}.$$

References: (Blum et al., 2018, Section 2.4.1), Wikipedia.

By Stirling's formula, we have the bound

$$\operatorname{vol} B(p,r) \leq \frac{\pi^{d/2} r^d}{(d/2e)^{d/2}} = \frac{(2e\pi)^{d/2} r^d}{d^{d/2}}. \tag{B.2.3}$$

The following fact allows us to compare norms.

Fact B.2.1. For all $x \in \mathbb{R}^d$,

$$\|x\|_p \leq \|x\|_r \leq d^{1/r-1/p} \cdot \|x\|_p \qquad \forall 1 \leq r \leq p \leq \infty.$$

In particular, the most useful cases are

$$\begin{aligned} \|x\|_1 &\geq \|x\|_2 \geq \|x\|_\infty \\ \frac{1}{\sqrt{d}} \|x\|_1 &\leq \|x\|_2 \leq \sqrt{d} \cdot \|x\|_\infty. \end{aligned}$$

References: Wikipedia.

Exercises

Exercise B.1. Although every norm satisfies the triangle inequality, the *squared* Euclidean norm does not. However, it does satisfy an *approximate* triangle inequality. Prove that, for any vectors u, v, we have

$$\|u - v\|_2^2 \;\leq\; 2 \cdot (\|u\|_2^2 + \|v\|_2^2).$$

B.3 Facts from Convex Analysis

B.3.1 One-dimensional functions

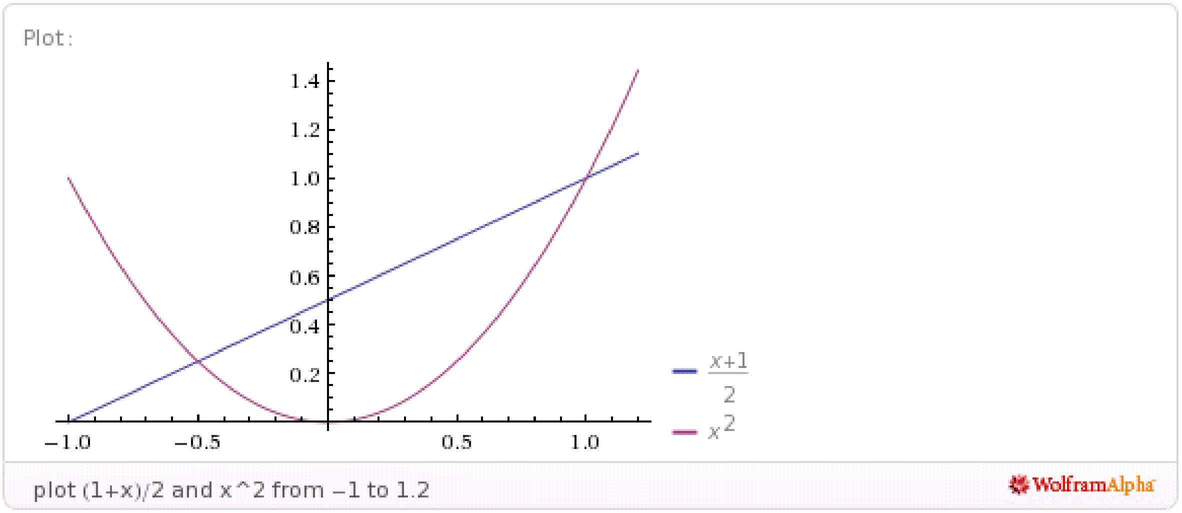
Let $f : S \to \mathbb{R}$ be a function defined on an interval $S \subseteq \mathbb{R}$.

Definition B.3.1. We say that f is convex on S if

$$f(\lambda x + (1-\lambda)y) \;\leq\; \lambda f(x) + (1-\lambda) f(y)$$

for all $x, y \in S$ and all $\lambda \in [0, 1]$.

Geometrically, this says that the secant line between the points $(x, f(x))$ and $(y, f(y))$ lies above the function f. The following example illustrates that $f(x) = x^2$ is convex.



An equivalent statement of this definition is as follows.

Fact B.3.2. Suppose $f : S \to \mathbb{R}$ is convex. Then, for any points $x, y \in S$ with $x < y$, we have

$$f(z) \;\leq\; \frac{f(y) - f(x)}{y - x} \cdot (z - x) + f(x)$$

for all $z \in [x, y]$.

For sufficiently nice functions, one can easily determine convexity by looking at its second derivative.

Fact B.3.3. Suppose $f : S \to \mathbb{R}$ is twice differentiable. Then f is convex if and only if the second derivative f'' is non-negative on the interior of S.

In our example above we used $f(x) = x^2$. Its second derivative is $f''(x) = 2$, which is non-negative, so f is convex.

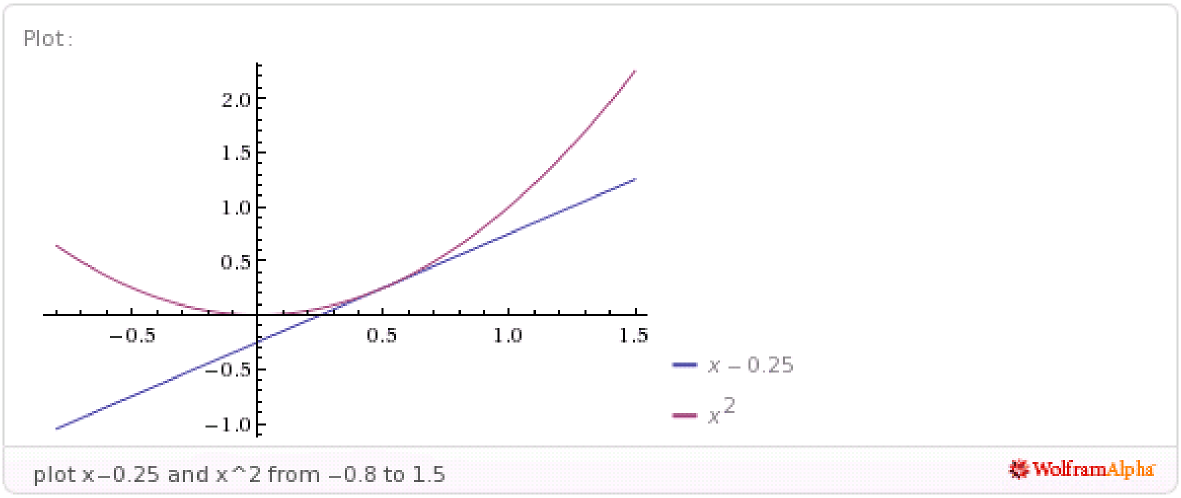
The next fact says that, for convex functions, the linear approximation at any point lies beneath the function.

Fact B.3.4 (Subgradient Inequality). Suppose $f : S \to \mathbb{R}$ is convex, and f is differentiable at a point $x \in S$. Then

$$f(y) \;\geq\; f(x) + f'(x)(y - x),$$

for any point $y \in S$.

The following example illustrates this for $f(x) = x^2$ at the point $x = 0.5$.



B.3.2 Various Inequalities from Convexity

In the analysis of randomized algorithms, we frequently use various inequalities to simplify expressions or make them easier to work with. These inequalities usually follow by convexity arguments and basic calculus. For example, let us revisit the following familiar fact.

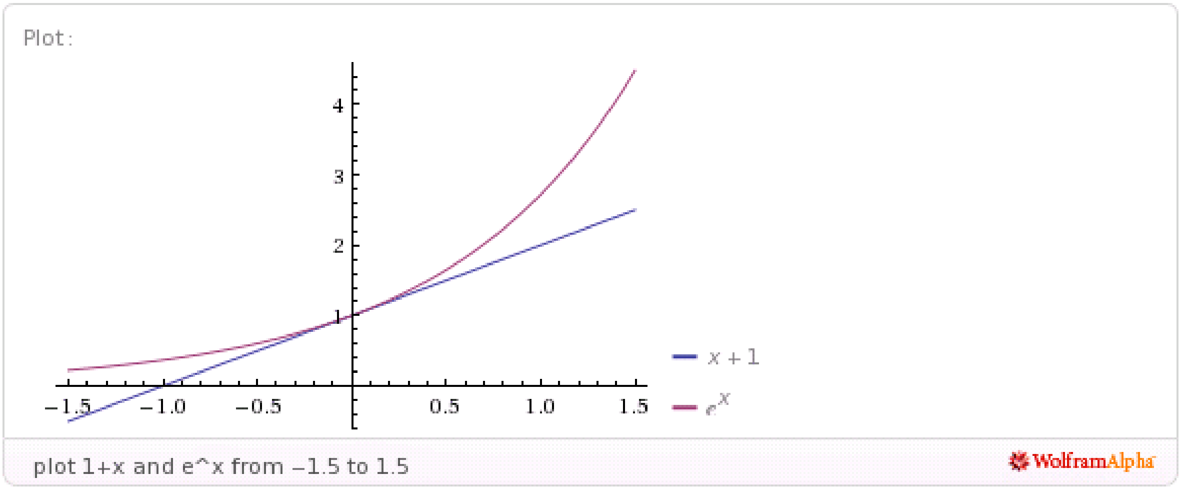
Fact A.2.5 (Approximating e^x near zero). For all real numbers x,

$$1 + x \leq e^x.$$

Moreover, for x close to zero, we have $1 + x \approx e^x$.

Proof. Convexity of e^x follows from Fact B.3.3 since its second derivative is e^x, which is non-negative on $\mathbb{R}$. Applying Fact B.3.4 at the point $x = 0$, we obtain

$$f(y) \;\geq\; f'(x) \cdot (y - x) + f(x) \;=\; y + 1. \qquad \square$$

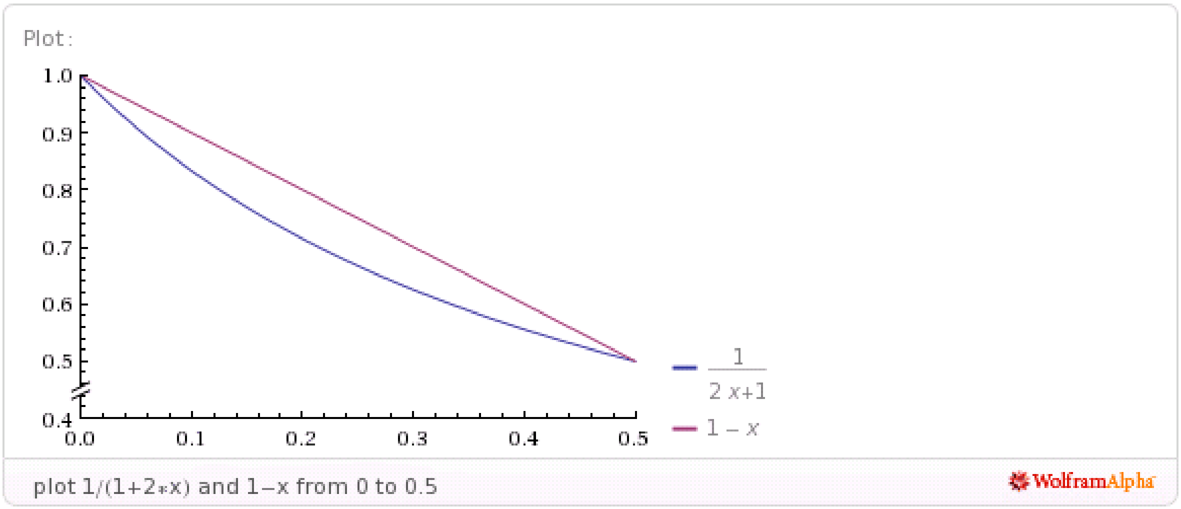


Fact B.3.5.

$$\frac{1}{1+2z} \;\leq\; 1-z \qquad \forall z \in [0,1/2].$$

Proof. Convexity of $f(z) = 1/(1+2z)$ on the set $S = (0,\infty)$ follows from Fact B.3.3 since its second derivative is $8/(1+2z)^3$, which is non-negative on S. Applying Fact B.3.2 at $x=0$ and $y=1/2$, we obtain

$$\begin{aligned} f(z) \;&\leq\; \frac{f(y)-f(x)}{y-x}\cdot(z-x)+f(x) \\ &=\; \frac{1/2-1}{1/2}\cdot(z-0)+1 \;=\; 1-z \qquad \forall z \in [0,1/2]. \end{aligned}$$ □



Fact B.3.6.

$$\log\frac{1}{1-x} \;\leq\; x+x^2 \qquad \forall x \in [0,1/2]$$

Proof. Observe that both sides equal 0 when $x=0$. The derivative of the left-hand side is $1/(1-x)$, whereas the derivative of the right-hand side is $1+2x$. We have $1/(1-x) \leq 1+2x$ for $x \in [0,1/2]$ by Fact B.3.5. Thus, by integrating, we obtain the desired inequality. □

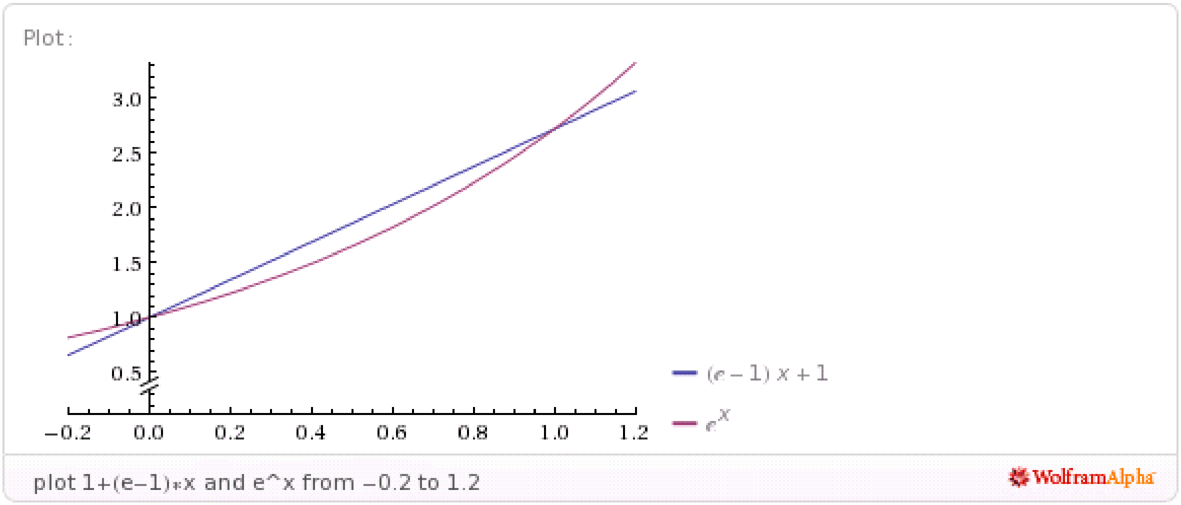
Fact B.3.7. $e^{-x} \leq 1 - x + \frac{x^2}{2}$ for $x \geq 0$.

Proof. Observe that both sides equal 0 when $x = 0$. The derivative of the left-hand side is $-e^{-x}$, whereas the derivative of the right-hand side is $-1 + x$. We have $-e^{-x} \leq -(1-x)$ by Fact A.2.5. Integrating proves the result. □

Fact B.3.8. Fix any $\alpha > 0$. Then $\alpha^z \leq 1 + (\alpha - 1)z$ for all $z \in [0,1]$.

Proof. The second derivative of α^z is $\alpha^z \cdot \ln^2(\alpha)$, which is non-negative, so α^z is convex. Applying Fact B.3.2 at the points $x = 0$ and $y = 1$, we obtain

$$\alpha^z \;\leq\; \frac{\alpha^y - \alpha^x}{y - x} \cdot (z - x) + \alpha^x \;=\; (\alpha - 1) \cdot z + 1. \qquad \square$$



Exercises

Exercise B.2. Prove that

$$\begin{aligned} \ln(1+z) \;&\leq\; z && \text{for } z > -1 \\ \ln(1+z) \;&\geq\; \ln(2)z \;\geq\; z/2 && \text{for } z \in [0,1] \end{aligned}$$

B.3.3 Multi-dimensional functions

Definition B.3.9 (Subgradient). Let $f : \mathcal{X} \to \mathbb{R}^n$ be a function. Recall that a ***subgradient*** of f at x is any vector g satisfying:

$$f(y) \;\geq\; f(x) + \langle\, g,\, y - x \,\rangle \qquad \forall y \in \mathcal{X}. \tag{B.3.1}$$

References: (Shalev-Shwartz and Ben-David, 2014, Definition 14.4).

Fact B.3.10 (Lipschitz equivalence). Let $\mathcal{X}$ be convex and open. Let $f : \mathcal{X} \to \mathbb{R}$ be convex. The following conditions are equivalent.

- $f : \mathcal{X} \to \mathbb{R}$ is L-Lipschitz:

$$|f(x) - f(y)| \;\leq\; L\,\|x-y\|_2 \qquad \forall x, y \in \mathcal{X}. \tag{B.3.2}$$

- f has bounded subgradients:

$$\|g\|_2 \leq L \qquad \forall w \in X,\; g \in \partial f(w). \tag{B.3.3}$$

References: (Shalev-Shwartz and Ben-David, 2014, Definition 12.6 and Lemma 14.7).

Fact B.3.11. Let $f_1, \ldots, f_n$ be convex functions, and let $\alpha_1, \ldots, \alpha_n \geq 0$. Then

$$\partial\big(\alpha_1 f_1(x) + \cdots + \alpha_n f_n(x)\big) \;=\; \alpha_1 \partial f_1(x) + \cdots \alpha_n \partial f_n(x).$$

The sum on the right-hand side is a Minkowski sum.

References: (Hiriart-Urruty and Lemaréchal, 2001, Theorem D.4.1.1).

Fact B.3.12 (Projection decreases Euclidean distance). $\|\Pi_{\mathcal{X}}(y) - x\|_2 \leq \|y - x\|_2$ for all $x \in \mathcal{X}$.

References: (Shalev-Shwartz and Ben-David, 2014, Lemma 14.9).

B.4 Probability

B.4.1 Expectation

For non-negative integer-valued random variables, the expectation has the formula Fact A.3.10. The following fact gives the analogous formula for arbitrary random variables.

Fact B.4.1. Let X be a non-negative random variable. Then

$$\mathrm{E}\,[\,X\,] \;=\; \int_0^\infty \Pr\,[\,X \geq x\,]\; dx \;=\; \int_0^\infty (1 - F(x))\, dx,$$

where F is the CDF of X.

References: For the special case of continuous random variables, see (Grimmett and Stirzaker, 2001, Lemma 4.3.4) or (Mitzenmacher and Upfal, 2005, Lemma 8.1). For arbitrary random variables, see (Vershynin, 2018, Lemma 1.2.1), (Durrett, 2019, Exercise 1.7.2) and (Klenke, 2008, Theorem 4.26).

Fact B.4.2 (Jensen's inequality). Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function and X a random vector.

$$\begin{aligned}\text{If } f \text{ is convex:}\quad & f(\mathrm{E}\,[\,X\,]) \;\leq\; \mathrm{E}\,[\,f(X)\,]\\ \text{If } f \text{ is concave:}\quad & f(\mathrm{E}\,[\,X\,]) \;\geq\; \mathrm{E}\,[\,f(X)\,]\end{aligned}$$

The following special case is useful. Let $x_1, \ldots, x_n \in \mathbb{R}^n$. Let $\lambda_1, \ldots, \lambda_n \in [0,1]$ satisfy $\sum_{i=1}^n \lambda_i = 1$. Then

$$\text{If } f \text{ is convex:}\quad f\Big(\sum_{i=1}^n \lambda_i x_i\Big) \;\leq\; \sum_{i=1}^n \lambda_i f(x_i).$$

References: (Cormen et al., 2001, Equation (C.26)), (Vershynin, 2018, page 7), (Mitzenmacher and Upfal, 2005, Lemma 2.4), (Klenke, 2008, Theorem 7.9), (Durrett, 2019, Theorem 1.5.1), (Grimmett and Stirzaker, 2001, Exercise 5.6.1), Wikipedia.

B.4.2 Variance

Definition B.4.3. The ***variance*** of a random variable X is defined to be

$$\mathrm{Var}\left[X \right] \;=\; \mathrm{E}\left[(X - \mathrm{E}\left[X \right])^2 \right].$$

References: (Lehman et al., 2018, Definition 20.2.2), (Cormen et al., 2001, Equation (C.27)), (Motwani and Raghavan, 1995, page 443), (Mitzenmacher and Upfal, 2005, Definition 3.2), (Grimmett and Stirzaker, 2001, Definition 3.3.5), (Durrett, 2019, page 29).

Fact B.4.4. Variance satisfies the following identity.

$$\mathrm{Var}\left[X \right] \;=\; \mathrm{E}\left[X^2 \right] - \mathrm{E}\left[X \right]^2. \tag{B.4.1}$$

Consequently,

$$\mathrm{Var}\left[X \right] \;=\; \mathrm{E}\left[X^2 \right] \qquad \text{if} \qquad \mathrm{E}\left[X \right] = 0. \tag{B.4.2}$$

References: (Lehman et al., 2018, Definition 20.3.1), (Anderson et al., 2017, Fact 3.48), (Cormen et al., 2001, Equation (C.27)), (Motwani and Raghavan, 1995, Proposition C.8), (Mitzenmacher and Upfal, 2005, Definition 3.2), (Grimmett and Stirzaker, 2001, page 51), (Durrett, 2019, Equation (1.6.2)), (Klenke, 2008, Definition 5.1(iii)).

Proof.

$$\begin{aligned}
\mathrm{Var}\left[X \right] &= \mathrm{E}\left[(X - \mathrm{E}\left[X \right])^2 \right] && \text{(by definition)}\\
&= \mathrm{E}\left[X^2 - 2X\,\mathrm{E}\left[X \right] + \mathrm{E}\left[X \right]^2 \right] && \text{(expand the quadratic)}\\
&= \mathrm{E}\left[X^2 \right] - 2\,\mathrm{E}\left[X \right]\mathrm{E}\left[X \right] + \mathrm{E}\left[X \right]^2 && \text{(linearity of expectation)}\\
&= \mathrm{E}\left[X^2 \right] - \mathrm{E}\left[X \right]^2. && \text{(simplifying).}
\end{aligned}$$

This proves (B.4.1). The equation (B.4.2) is immediate. □

Fact B.4.5. Let $G_1, \ldots, G_d$ be *pairwise* independent random variables with finite variance. Let $\sigma_1, \ldots, \sigma_d \in \mathbb{R}$ be arbitrary. Then $\mathrm{Var}\left[\sum_{i=1}^d \sigma_i G_i \right] = \sum_{i=1}^d \sigma_i^2 \,\mathrm{Var}\left[G_i \right]$.

References: (Lehman et al., 2018, Lemma 20.3.4 and Lemma 20.3.8), (Anderson et al., 2017, Facts 3.52 and 8.11), (Cormen et al., 2001, page 1200), (Mitzenmacher and Upfal, 2005, Theorem 3.5 and Exercise 3.4), (Grimmett and Stirzaker, 2001, Theorem 3.3.11).

B.4.3 Gaussian random variables

One of the most important continuous distributions is the ***Gaussian distribution***. It is also called the ***Normal distribution***. This distribution has two real parameters, its mean (denoted μ) and its variance (denoted σ^2). The distribution with those parameters is denoted $N(\mu, \sigma^2)$.

References: (Anderson et al., 2017, Definition 3.60), (Grimmett and Stirzaker, 2001, Definition 4.4.4), Wikipedia.

One useful property is that sums of Gaussians are also Gaussian.

Fact B.4.6. Let $g_1, \ldots, g_d$ be independent random variables where g_i has distribution $N(0,1)$. Then, for any scalars $\sigma_1, \ldots, \sigma_d$, the sum $\sum_{i=1}^d \sigma_i g_i$ has distribution $N(0, \sum_{i=1}^d \sigma_i^2) = N(0, \|\sigma\|_2^2)$.

References: (Anderson et al., 2017, Example 7.8 and Example 8.20), (Grimmett and Stirzaker, 2001, Example 4.8.3), (Durrett, 2019, Theorem 2.1.20 and Corollary 3.3.13), Wikipedia.

Remark B.4.7. Fact B.4.6 can be viewed in the more abstract context of ***stable random variables***. The Gaussian distribution is 2-stable. More generally, if X is an α-stable RV, and $X_1, \ldots, X_d$ are independent copies of X, then $\sum_{i=1}^{d} \sigma_i X_i$ has the distribution $\|\sigma\|_\alpha X$. See Equation (1.8) in "Stable Distributions: Models for Heavy Tailed Data" with $\beta = 0$, $\gamma_i = \sigma_i$ and $\delta = 0$.

References: (Durrett, 2019, Section 3.8), Wikipedia.

A useful fact about the Gaussian distribution is the following bound on its right tail.

Fact B.4.8 (Gaussian tail bound). Let X have the distribution $N(0,1)$. Let $x > 0$. Then

$$\frac{1}{\sqrt{2\pi}}(x^{-1} - x^{-3}) \exp(-x^2/2) \;\leq\; \Pr[\, X \geq x \,] \;\leq\; \frac{1}{\sqrt{2\pi}} x^{-1} \exp(-x^2/2).$$

References: (Vershynin, 2018, Proposition 2.1.2), (Durrett, 2019, Theorem 1.2.6), (Feller, 1968, Lemma VII.1.2), (Wainwright, 2019, Exercise 2.2).

Bibliography

Alon, N. and Spencer, J. (2000). *The probabilistic method*. Wiley Interscience, second edition.

Anderson, D. F., Sepäläinen, T., and Valkó, B. (2017). *Introduction to Probability*. Cambridge.

Arora, S., Hazan, E., and Kale, S. (2012). The multiplicative weights update method: A meta algorithm and its applications. *Theory of Computing*, 8(6):121–164.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. (2002). The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. (1995). Gambling in a rigged casino: The adversarial multi-arm bandit problem. In *36th Annual Symposium on Foundations of Computer Science*, pages 322–331.

Blum, A., Hopcroft, J., and Kannan, R. (2018). Foundations of data science.
https://www.cs.cornell.edu/jeh/book.pdf.

Boucheron, S., Lugosi, G., and Massart, P. (2012). *Concentration Inequalities: A nonasymptotic theory of independence*. Oxford.

Bubeck, S. (2015). Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8(3-4):231–357.
https://arxiv.org/abs/1405.4980.

Cesa-Bianchi, N. and Lugosi, G. (2006). *Prediction, learning, and games*. Cambridge University Press.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms*. MIT Press, Cambridge, MA, second edition.

Dubhashi, D. P. and Panconesi, A. (2009). *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press.

Durrett, R. (2019). *Probability: Theory and Examples*. Cambridge, fifth edition.
https://services.math.duke.edu/~rtd/PTE/PTE5_011119.pdf.

Feller, W. (1968). *An Introduction to Probability Theory and Its Applications, Volume I*. John Wiley & Sons, third edition.

Grigoriadis, M. and Khachiyan, L. (1995). A sublinear-time randomized approximation algorithm for matrix games. *Operations Research Letters*, 18:53–58.

Grimmett, G. and Stirzaker, D. (2001). *Probability and Random Processes*. Oxford University Press, third edition.

Guruswami, V., Rudra, A., and Sudan, M. (2019). Essential coding theory.
https://cse.buffalo.edu/faculty/atri/courses/coding-theory/book/.

Hazan, E. (2015). Introduction to online convex optimization. *Foundations and Trends in Optimization*, 2(3–4).

Hiriart-Urruty, J.-B. and Lemaréchal, C. (1996). *Convex Analysis and Minimization Algorithms I: Fundamentals*. Springer.

Hiriart-Urruty, J.-B. and Lemaréchal, C. (2001). *Fundamentals of Convex Analysis*. Springer-Verlag.

Klenke, A. (2008). *Probability Theory: A Comprehensive Course*. Springer.

Lehman, E., Leighton, F. T., and Meyer, A. R. (2018). Mathematics for computer science.
https://courses.csail.mit.edu/6.042/spring18/mcs.pdf.

McDiarmid, C. (1998). Concentration.
http://cgm.cs.mcgill.ca/~breed/conc/colin.pdf.

Mitzenmacher, M. and Upfal, E. (2005). *Probability and computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press.

Motwani, R. and Raghavan, P. (1995). *Randomized Algorithms*. Cambridge University Press.

Murphy, K. P. (2022). *Probabilistic Machine Learning: An Introduction*. MIT Press.
https://probml.github.io/pml-book/book1.html.

Plotkin, S. A., Shmoys, D. B., and Tardos, É. (1995). Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research*, 20(2):257–301.

Roch, S. (2020). Modern discrete probability: An essential toolkit.
https://people.math.wisc.edu/~roch/mdp/index.html.

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press.
https://www.cs.huji.ac.il/w~shais/UnderstandingMachineLearning/index.html.

Shmoys, D. P. and Shmoys, D. B. (2010). *The Design of Approximation Algorithms*. Cambridge University Press.
http://designofapproxalgs.com/book.pdf.

Trefethen, L. N. and Bau, III, D. (1997). *Numerical Linear Algebra*. SIAM.

Vadhan, S. P. (2012). Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1–3):1–336.

Vershynin, R. (2018). *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge University Press.

Wainwright, M. J. (2019). *High-Dimensional Statistics: A Non-Asymptotic Viewpoint*. Cambridge.