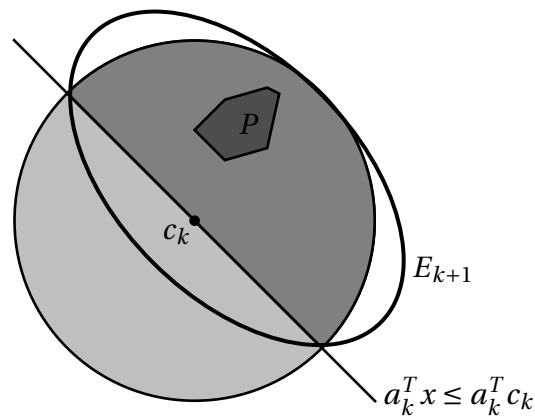


Design and Analysis of Algorithms

CSE 521 — Fall 2024

Thomas Rothvoss



UNIVERSITY *of*
WASHINGTON

Last changes: November 3, 2024

Contents

1	Randomized algorithms	7
1.1	Karger's algorithm	7
1.2	The Karger-Stein algorithm	11
1.3	Probability Theory	12
1.3.1	Markov's inequality	13
1.3.2	Union bound	13
1.3.3	Application: Fix points of permutations	14
1.3.4	Chebyshev's Inequality	14
1.3.5	Polling	16
1.3.6	The birthday paradox	17
1.3.7	Hoeffding Inequality	18
1.3.8	Polling	20
1.3.9	Random walk on a line	21
1.4	Discrepancy theory	21
1.5	Hashing	22
1.6	Limited independence	24
1.6.1	The birthday paradox revisited	25
1.6.2	Double Hashing	26
1.6.3	Construction of pairwise independent hash functions	27
1.7	Unbiased estimators	29
1.8	Streaming	32
2	The curse of dimensionality and dimension reduction	37
2.1	The nearest neighbor problem	37
2.1.1	Locally sensitive hash functions	39
2.2	Volume in higher dimensions	42
2.3	Nearly orthogonal vectors	44
2.4	Introductions to Gaussians	45
2.5	More on rotationally invariant distributions	46
2.6	Concentration of measure for Gaussians	47

2.7	Dimension reduction	49
3	Algebraic algorithms	53
3.1	Matrix Identity testing	55
3.2	The Schwarz Zippel Lemma	56
3.3	Bipartite matchings	57
3.4	Perfect matchings in general graphs	61
4	Linear algebra	63
4.1	Eigenvalues	63
4.1.1	The Spectral Theorem	64
4.1.2	Positive semidefinite matrices	64
4.1.3	A geometric interpretation	65
4.1.4	Applying functions to matrices	66
4.1.5	Trace, determinant and rank	67
4.1.6	Raleigh Quotient	69
4.2	The Singular Value Decomposition	70
4.3	Matrix norms	71
4.4	Best low rank approximation	72
4.5	Hidden Partition	75
4.6	Additive approximations for MaxCut	77
5	Spectral graph theory	83
5.1	Graph partitioning	85
5.2	Cheeger's Inequality	86
5.3	The power method	91
6	Linear programming	95
6.1	Convexity, polyhedra and linear programs	95
6.2	An overview over algorithms for linear programs	96
6.3	The ellipsoid method	98
6.3.1	The ellipsoid method with a separation oracle	102
6.4	Convex programming	105
6.5	Semidefinite programming	108
6.6	Rounding linear programs and SDPs	109
6.6.1	Vertex Cover	109
6.6.2	Set Cover	111
6.6.3	MaxCut	113
6.7	LP duality	115
6.8	An application to Nash Equilibria in Zero Sum Games	117

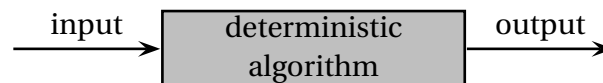
<i>CONTENTS</i>	5
7 Submodular functions	121
7.1 Maximizing a monotone submodular functions with a cardinality constraint	122
7.2 Application to the Target Set Selection Problem	124
A Notation and useful facts	131

Chapter 1

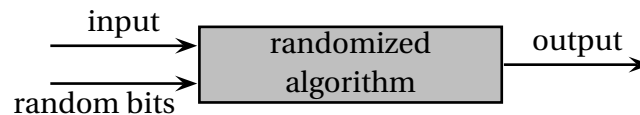
Randomized algorithms

These course notes are mainly based on the excellent lecture notes by Shayan Oveis Gharan from previous iterations of the same course. We will otherwise give references to the original work in place.

A *deterministic algorithm* takes an input instance and produces an output.



In contrast, a *randomized algorithm* depends on an input and *random bits* and produces an output.



In particular the same algorithm given the same instance might produce different outputs dependent on the random bits. It may seem that a deterministic algorithm is always preferable but in many cases the best known randomized algorithm is faster and/or simpler than the deterministic counterpart¹.

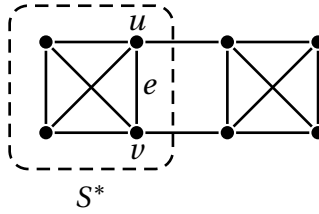
1.1 Karger's algorithm

The first algorithm that we discuss is due to Karger [Kar93]. For an undirected graph $G = (V, E)$ and $S \subseteq V$ we write $\delta(S) := \{e \in E \mid |e \cap S| = 1\}$ as all the edges

¹From a complexity-theoretic point of view it is unclear whether there is more than a polynomial factor difference between the running time of the best deterministic and the best randomized algorithm for any problem. In particular it is open whether the complexity classes **BPP** and **P** are identical or not. See CSE 531 for more background.

that have one endpoint in S and one endpoint in the complement $V \setminus S$. For the purpose of this section we allow that G is a *multigraph*, that means there may be several parallel edges between the same two vertices. But we disallow self-loops. We abbreviate $n := |V|$ as the number of vertices.

We are interested in the fundamental graph problem of finding the *mincut*, i.e. a set S with $\emptyset \subset S \subset V$ that minimizes $|\delta(S)|$. In other words, we want to find a cut that separates as few edges as possible. Formally speaking the edge set $\delta(S)$ is the *cut* but we will be sloppy and also talk about S as the cut.

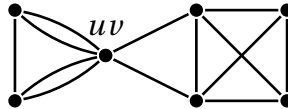


mincut S^* in a graph

First a formal definition:

Definition 1.1. Given an undirected (multi) graph $G = (V, E)$ and an edge $e = \{u, v\} \in E$. *Contracting* the edge e results in a graph where the vertices u, v are replaced by a new super-node uv and all edges previously incident to either u or v are incident to uv . We remove the created self-loop arising from e .

Note that contraction can create parallel edges. For example contracting the edge e in the graph depicted above gives:



Fix a set S^* s.t. $\delta(S^*)$ is a mincut. The idea of Karger's algorithm is as follows: the set $\delta(S^*)$ is the mincut and hence has few edges. So if we draw a uniform random edge $e \sim E$ then most likely $e \notin \delta(S^*)$. In that case we can contract the edge, and obtain a graph that is smaller by one node and still has the same mincut. So we made progress by doing very little work.

Lemma 1.2. Let $G = (V, E)$ be an undirected graph and let S^* be a mincut. Let $e \sim E$ be a uniform random edge. Then $\Pr[e \in \delta(S^*)] \leq \frac{2}{n}$.

Proof. Let $k := |\delta(S^*)|$ be the number of edges in the mincut and let $d(v) := |\delta(\{v\})|$ be the degree of vertex $v \in V$. Then

$$kn \stackrel{(*)}{\leq} \sum_{v \in V} \underbrace{d(v)}_{\geq k} \stackrel{(**)}{=} 2|E|$$

using in (*) that also each singleton $\{v\}$ is a cut. The equation in (**) is also called *Handshake lemma* and it follows from realizing that every edge $e \in E$ contributes 2 to the left hand side and right hand side of (**). Then

$$\Pr[e \in \delta(S^*)] = \frac{|\delta(S^*)|}{|E|} \leq \frac{k}{kn/2} = \frac{2}{n}$$

□

The full algorithm is as follows:

KARGER'S ALGORITHM
Input: Graph $G = (V, E)$
Output: Cut $\delta(S)$

- (1) FOR $i = 1$ TO $n - 2$ DO
 - (2) Pick a uniform edge $e \sim E$ and contract it.
- (3) Remaining V correspond to 2 supernodes. Return edges between the supernodes.

We will now analyze the algorithm:

Fact 1.3. For any graph $G = (V, E)$, contracting any edge does not decrease the size of the mincut.

We will leave this fact as homework².

Theorem 1.4. Let $G = (V, E)$ be a graph and let $\delta(S^*)$ be a mincut. Then the probability that the algorithm returns $\delta(S^*)$ is at least $\frac{2}{n(n-1)}$.

Proof. We fix a mincut $\delta(S^*)$ and abbreviate $k := |\delta(S^*)|$. Let A_i be the event that in step $i \in \{1, \dots, n-2\}$, the contracted edge is not in $\delta(S^*)$. Consider the situation in iteration i assuming the algorithm has not made a mistake so far, i.e. we condition on A_1, \dots, A_{i-1} to have happened. Then $\delta(S^*)$ still exists in the graph and by Fact 1.3 it still is a mincut. The graph at the beginning of the i th iteration has $n - i + 1$ vertices. Then applying Lemma 1.2 we have

$$\Pr[A_i \mid A_1, \dots, A_{i-1}] \geq 1 - \frac{2}{n - i + 1}$$

²In mathematics, a “fact” is the terminology for a theorem that the author is too lazy to prove.

Then the probability that the algorithm never makes a mistake is

$$\begin{aligned}
 \Pr[A_1, \dots, A_{n-2}] &= \prod_{i=1}^{n-2} \Pr[A_i \mid A_1, \dots, A_{i-1}] \\
 &\geq \prod_{i=1}^{n-2} \left(1 - \frac{2}{n-i+1}\right) \\
 &= \prod_{i=1}^{n-2} \frac{n-i-1}{n-i+1} \\
 &= \frac{\cancel{n-2} \cdot \cancel{n-3} \cdot \cancel{n-4} \cdots \cancel{3} \cdot 2 \cdot 1}{n \cdot n-1 \cdot \cancel{n-2} \cdots \cancel{3} \cdot \cancel{4} \cdot \cancel{3}} = \frac{2}{n(n-1)}
 \end{aligned}$$

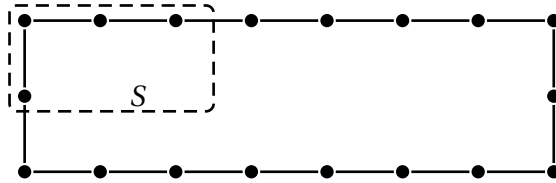
where we use Bayes' rule in the first step. In the last step we use that almost all products cancel out. \square

The algorithm by Karger has a very important combinatorial consequence:

Corollary 1.5. The number of different mincuts in a graph $G = (V, E)$ is at most $\binom{n}{2}$.

Proof. Let us reconsider the claim of Theorem 1.4. It does not just say that the probability to return *some* mincut is $\frac{2}{n(n-1)}$ but it says that the probability to return the *particular mincut* $\delta(S^*)$ is that high. Hence there can be at most $\frac{n(n-1)}{2} = \frac{1}{\binom{n}{2}}$ many mincuts. \square

This bound is tight. For example if G is a *cycle* on n vertices, then any selection of 2 edges forms a mincut.



one of $\binom{n}{2}$ many mincuts in an n -node cycle

So the success probability of Karger's algorithm is only $\Theta(\frac{1}{n^2})$. But we could repeat the algorithm $O(n^2 \log(n))$ times and return the best found cut and boost the probability to find a mincut to at least $1 - \frac{1}{n}$. We can formulate this in more general terms:

Lemma 1.6. Consider a randomized algorithm A that has success probability at least $0 < p < 1$. Repeating the algorithm $\ln(\frac{1}{\delta})/p$ times, the probability that the algorithm has success at least once is at least $1 - \delta$.

Proof. Set $k := \ln(\frac{1}{\delta})/p$. Then

$$\Pr[\text{no success in } k \text{ repetitions}] \leq (1-p)^k \leq \exp(-pk) = \exp\left(-\ln\left(\frac{1}{\delta}\right)\right) = \delta$$

Here we use that $1+x \leq e^x$ for all $x \in \mathbb{R}$. □

One can run one iteration of Karger's algorithm in time $O(n^2)$. Then with $O(n^2 \log(n))$ repetitions we have an running time of $O(n^4 \log(n))$ to reach a success probability of $1 - \frac{1}{n}$. However, we can improve that running time significantly which is a result due to Karger and Stein [KS93].

1.2 The Karger-Stein algorithm

The crucial insight in order to improve the running time is the following: Karger's algorithm is much more likely to make a mistake at the end when only few nodes are left than at the beginning when we still have many nodes. So one only needs to boost the success probability towards the end of the algorithm. We turn this into a recursive algorithm where we contract $(1-\alpha)n$ edges at random, then recurse *twice*. Here $0 < \alpha < 1$ is a constant parameter that we determine later.

KARGER-STEIN ALGORITHM

Input: Graph $G = (V, E)$

Output: Cut $\delta(S)$

- (1) If $n := |V| = 2$ then return the unique cut
- (2) FOR $i = 1$ TO $(1-\alpha)n$ DO
 - (3) Pick a uniform edge $e \sim E$ and contract it.
- (4) Let G' be the contracted graph.
- (5) Call KARGERSTEIN(G') twice and return the best of both cuts.

It remains to analyze the running time and the success probability.

Lemma 1.7. For any constant $0 < \alpha \leq \frac{1}{\sqrt{2}}$ the algorithm has running time of $O(n^2 \log n)$.

Proof. The work done in the loop (2)+(3) takes time at most $O(n^2)$ and the graph G' has αn vertices. So the running time satisfies the recursion

$$T(n) \leq Cn^2 + 2 \cdot T(\alpha n)$$

We could look up CSE 421 how to resolve such a recursion, but we give a self-contained argument. Consider the j -th level of the recursion for $j \geq 0$. There

will be 2^j calls at this recursion level and the graph at this level has size $n \cdot \alpha^j$. So the total amount of work done at recursion level j is $2^j \cdot C \cdot (n \cdot \alpha^j)^2$. So in order to keep the work per level at $O(n^2)$ we need $2\alpha^2 \leq 1$ which means one should choose $\alpha \leq \frac{1}{\sqrt{2}}$. The number of levels will be³ $O(\log n)$. \square

We have learned that from now on we should set $\alpha := \frac{1}{\sqrt{2}}$.

Theorem 1.8. *The probability that the algorithm finds a mincut is at least $\frac{1}{2 \log_2(n)}$.*

Proof. We fix a mincut $\delta(S^*)$. Reusing the notation from Theorem 1.4 we can lower bound the probability of *not* contracting an edge inside $\delta(S^*)$ during the loop (2)+(3) by⁴

$$\Pr[A_1, \dots, A_{(1-\alpha)n}] \geq \prod_{i=1}^{(1-\alpha)n} \frac{n-i-1}{n-i+1} = \frac{(\alpha n - 1) \cdot \alpha n}{n(n-1)} \approx \frac{(\alpha n)^2}{n^2} \stackrel{\alpha = \frac{1}{\sqrt{2}}}{=} \frac{1}{2}$$

As the algorithm is recursive, it will be useful to complete the proof with an induction over the number of recursions. Clearly the algorithm terminates on an n -node instance with $O(\log n)$ recursions. Let $p(r)$ be the minimum success probability over all instances where the algorithm takes r -levels of recursion. We prove by induction over $r \in \mathbb{Z}_{\geq 0}$ that $p(r) \geq \frac{1}{2(r+1)}$. If $r = 0$, then the argument above shows that $p(0) \geq \frac{1}{2}$. Now suppose $r \geq 1$. Then the Karger-Stein algorithm will succeed if no edge in $\delta(S^*)$ is contracted and at least one of the recursive calls is successful. Hence

$$p(r) \geq \frac{1}{2} \cdot \underbrace{\left(1 - (1 - p(r-1))^2\right)}_{\Pr[\text{both rec. calls fail}]} = p(r-1) - \frac{1}{2}p(r-1)^2 \geq \frac{1}{2r} - \frac{1}{8r^2} \geq \frac{1}{2(r+1)}$$

Here in the second inequality we use that the function $x \mapsto x - \frac{1}{2}x^2$ is monotonically increasing on the interval $[0, 1]$. \square

1.3 Probability Theory

Much of the work in analyzing randomized algorithms deals with understanding how the random variables related to the algorithm behave. We will discuss some tools to analyze random variables and provide some toy applications.

³The implicit constant will depend on α .

⁴Well, we cheated here. Actually the inequality goes into the wrong direction and it only gives that $\Pr[A_1, \dots, A_{(1-\alpha)n}] \geq \frac{1}{2} - \Theta(\frac{1}{n})$. One can still make the analysis work, but it does get more cumbersome.

1.3.1 Markov's inequality

Suppose you learn that the average home value in Seattle is \$1,000,000 and you would like to estimate what fraction of homes are worth more than \$3,000,000. Certainly that fraction can be at most $1/3$ since otherwise the average would be higher already due to those pricey homes. One might get the feeling that this is a very loose estimate but if indeed the only information that one has on the home values is the average, then this is the best possible estimate. This simple argument is called *Markov's inequality*.

Theorem 1.9. For any random variable $X \geq 0$ and any $t > 0$ one has $\Pr[X \geq t \cdot \mathbb{E}[X]] \leq \frac{1}{t}$.

Proof. We set $s := t\mathbb{E}[X]$ and prove that $\Pr[X \geq s] \leq \frac{\mathbb{E}[X]}{s}$. By the law of total expectation

$$\mathbb{E}[X] = \underbrace{\mathbb{E}[X \mid X \geq s]}_{\geq s} \cdot \Pr[X \geq s] + \underbrace{\mathbb{E}[X \mid X < s]}_{\geq 0} \cdot \Pr[X < s] \geq s \cdot \Pr[X \geq s]$$

Rearranging gives the claim. □

The bound is tight for example if

$$X = \begin{cases} t & \text{with probability } \frac{1}{t} \\ 0 & \text{otherwise} \end{cases}$$

for $t \geq 1$. But Markov's inequality is extremely useful as it only requires knowing the expectation.

1.3.2 Union bound

Given a list of events E_1, \dots, E_n , the probability that at least one of them happens is at most the sum of their probabilities. This simple fact is called the *union bound*:

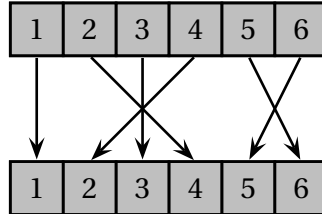
Lemma 1.10 (Union bound). For any events E_1, \dots, E_m one has $\Pr[E_1 \cup \dots \cup E_m] \leq \sum_{i=1}^m \Pr[E_i]$.

The bound holds with equality if all the events are disjoint. Set-theoretically this statement is even more obvious: for any family S_1, \dots, S_m of finite sets one has

$$|S_1 \cup \dots \cup S_m| \leq \sum_{i=1}^m |S_i|$$

1.3.3 Application: Fix points of permutations

We abbreviate $[n] := \{1, \dots, n\}$ where $n \in \mathbb{N}$. A bijective map $\sigma : [n] \rightarrow [n]$ is called a *permutation*. An element $i \in [n]$ is called a *fix point* of permutation σ if $\sigma(i) = i$.



permutation σ with fix points at 1 and 3

Because of dependencies, analyzing the distribution of fix points of a random permutation is not easy. But we can give the following simple estimate:

Lemma 1.11. *Draw a uniform random permutation $\sigma : [n] \rightarrow [n]$ and let X be the number of its fix points. Then for any $t \geq 1$ one has*

$$\Pr[X \geq t] \leq \frac{1}{t}$$

Proof. Let $X_i := \mathbf{1}_{\sigma_i=i}$, i.e. $X_i \in \{0, 1\}$ is the *indicator random variable* that tells us whether i is a fix point. Note that $X = \sum_{i=1}^n X_i$. Clearly $\Pr[X_i = 1] = \frac{1}{n}$. Then

$$\mathbb{E}[X] = \mathbb{E}\left[\sum_{i=1}^n X_i\right] \stackrel{(*)}{=} \sum_{i=1}^n \underbrace{\mathbb{E}[X_i]}_{=1/n} = 1$$

Here the rule used in (*) is called *linearity of expectation* and it holds even though the random variables X_1, \dots, X_n are not independent. Then by Markov inequality $\Pr[X \geq t] \leq \frac{1}{t}$. \square

1.3.4 Chebyshev's Inequality

Next we will see an inequality that (often) allows better concentration of a random variable.

Definition 1.12. For a random variable X , the *variance* is

$$\text{Var}[X] := \mathbb{E}[(X - \mathbb{E}[X])^2]$$

We prove a simple identity for the variance:

Lemma 1.13. For any random variable, $\text{Var}[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$.

Proof. We abbreviate $\mu := \mathbb{E}[X]$. Then using linearity of expectation

$$\text{Var}[X] = \mathbb{E}[(X - \mu)^2] = \mathbb{E}[X^2] - 2\mu \underbrace{\mathbb{E}[X]}_{=\mu} + \mu^2 = \mathbb{E}[X^2] - \mu^2$$

□

By definition we know that $\text{Var}[X] \geq 0$ for any random variable. Then by Lemma 1.13 we know that⁵ $\mathbb{E}[X^2] \geq \mathbb{E}[X]^2$.

Theorem 1.14 (Chebyshev's Inequality). For any random variable X and any $s > 0$ one has

$$\Pr[|X - \mathbb{E}[X]| \geq s] \leq \frac{\text{Var}[X]}{s^2}$$

Equivalently, for any $t > 0$ one has

$$\Pr[|X - \mathbb{E}[X]| \geq t\sigma] \leq \frac{1}{t^2}$$

where $\sigma := \sqrt{\text{Var}[X]}$ is the standard deviation.

Proof. The proof works by just applying Markov's inequality to the squared deviation. Set $Y := (X - \mu)^2$ where $\mu := \mathbb{E}[X]$. Then

$$\Pr[|X - \mu| \geq s] = \Pr[Y \geq s^2] \leq \frac{\mathbb{E}[Y]}{s^2} = \frac{\text{Var}[X]}{s^2}$$

□

Of course in order to effectively use Chebyshev's Inequality we would need to be able to argue that the variance of a random variable is small. The easiest argument is the following. Recall that discrete⁶ random variables X_1, \dots, X_n are *pairwise independent* if for all distinct $i, j \in [n]$ and any s, t one has $\Pr[X_i = s, X_j = t] = \Pr[X_i = s] \cdot \Pr[X_j = t]$.

⁵There is a more general approach to derive why this must hold. The function $f : \mathbb{R} \rightarrow \mathbb{R}$ with $f(x) := x^2$ is *convex*. Then *Jensen's inequality* says that for any convex function f and any random variable X one has $\mathbb{E}[f(X)] \geq f(\mathbb{E}[X])$.

⁶A *discrete* random variable is one whose support is countable. For continuous random variables one needs to use the density function rather than probability of individual values but the spirit of the definition is the same.

Lemma 1.15. For any set X_1, \dots, X_n of pairwise independent random variables one has

$$\text{Var}\left[\sum_{i=1}^n X_i\right] = \sum_{i=1}^n \text{Var}[X_i]$$

Proof. We abbreviate $\mu_i := \mathbb{E}[X_i]$. Then multiplying out and using linearity of expectation gives

$$\begin{aligned} \text{Var}\left[\sum_{i=1}^n X_i\right] &= \mathbb{E}\left[\left(\sum_{i=1}^n X_i\right)^2\right] - \left(\sum_{i=1}^n \mu_i\right)^2 \\ &= \sum_{i=1}^n (\mathbb{E}[X_i^2] - \mu_i^2) + \sum_{i \neq j} \underbrace{(\mathbb{E}[X_i X_j] - \mu_i \mu_j)}_{=0} = \sum_{i=1}^n \text{Var}[X_i] \end{aligned}$$

Here we use that $\mathbb{E}[X_i X_j] = \mathbb{E}[X_i] \mathbb{E}[X_j]$ due to pairwise independence. \square

1.3.5 Polling

Suppose we have an unknown distribution \mathcal{D} over numbers \mathbb{R} and we would like to estimate its mean $\mu := \mathbb{E}_{x \sim \mathcal{D}}[x]$. We can draw independent random samples

$$X_1, \dots, X_n \sim \mathcal{D}$$

and be optimistic that the *empirical mean* $Y := \frac{1}{n} \sum_{i=1}^n X_i$ would be close to the expectation μ . But how many samples do we need to be relatively certain that the error is at most say $\varepsilon > 0$? By some abuse of notation we write $\text{Var}[\mathcal{D}]$ as the variance of a random variable that is drawn from \mathcal{D} .

Theorem 1.16. Let Y be the empirical mean of n samples from \mathcal{D} . Let μ be the mean of \mathcal{D} .

- (i) For any $\varepsilon > 0$ one has $\Pr[|Y - \mu| \geq \varepsilon] \leq \frac{\text{Var}[\mathcal{D}]}{n\varepsilon^2}$.
- (ii) If \mathcal{D} is a distribution over $\{0, 1\}$, then $\Pr[|Y - \mu| \geq \mu] \leq \frac{1}{4n\varepsilon^2}$.

Proof. The variance of the empirical mean is

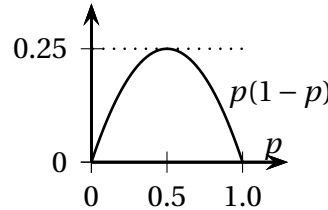
$$\text{Var}[Y] = \text{Var}\left[\frac{1}{n} \sum_{i=1}^n X_i\right] = \frac{1}{n^2} \sum_{i=1}^n \underbrace{\text{Var}[X_i]}_{=\text{Var}[\mathcal{D}]} = \frac{1}{n} \cdot \text{Var}[\mathcal{D}]$$

Then Chebychev's inequality gives (i). For (ii), consider a random variable $X \sim \mathcal{D}$ where we know that $X \in \{0, 1\}$. That means for some $0 \leq p \leq 1$ one has

$$X = \begin{cases} 1 & \text{probability } p \\ 0 & \text{probability } 1 - p \end{cases}$$

Then

$$\text{Var}[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2 = 1^2 \cdot p + 0^2 \cdot (1-p) - p^2 = p(1-p)$$



One can then verify that $p(1-p) \leq \frac{1}{4}$ (which is attained for $p = \frac{1}{2}$). \square

One could easily generalize (ii) to distributions \mathcal{D} that are supported on $[0, 1]$ or more generally on intervals $[a, b]$. It is straightforward to get a bound of $\frac{|a-b|^2}{n\epsilon^2}$ in the latter case and which a bit more work one can show the tight bound of $\frac{|a-b|^2}{4n\epsilon^2}$.

1.3.6 The birthday paradox

The *birthday paradox* is the following popular statement: *how many people in a room does one need until one could expect that at least two of them have the same birthday? The answer is 23!* This seems counterintuitive as one might have expected the answer to be closer to $\frac{1}{2} \cdot 365$; we note that instead $23 \approx \sqrt{365}$ (well, at least in terms of ballpark). The principle is important for the purpose of this class as it corresponds to the chance that a random function has collisions.

Suppose we have a universe $\{1, \dots, m\}$ and we draw independently and uniformly at random $X_1, \dots, X_n \sim \{1, \dots, m\}$. We say that there is a *collision* if there are indices $i \neq j$ so that $X_i = X_j$. Then for the birthday paradox we have $m = 365$ and want to know how large n needs to be so that the probability of having a collision exceeds $1/2$. We can now prove that the threshold is roughly $n \approx \sqrt{m}$.

Theorem 1.17. *Let $X_1, \dots, X_n \sim \{1, \dots, m\}$ independently at random. Then*

(i) *If $n \leq \sqrt{m}$, then $\Pr[\text{no collision}] \geq \frac{1}{2}$.*

(ii) *If $n \geq c\sqrt{m}$ for $c > \sqrt{2}$ then $\Pr[\text{no collision}] \leq \frac{2}{c^2} + o(1)$.*

Proof. First we prove (i). For indices $1 \leq i < j \leq n$ we let $Y_{ij} := \mathbf{1}_{X_i = X_j}$ be the indicator random variable telling whether there is a collision for the pair i, j . Let $Y := \sum_{1 \leq i < j \leq n} Y_{ij}$ be the total number of pairwise collisions. Note that we have

no collision if and only if $Y = 0$. Using linearity of expectation we have

$$\mathbb{E}[Y] = \sum_{1 \leq i < j \leq n} \underbrace{\mathbb{E}[Y_{ij}]}_{=\Pr[Y_{ij}=1]=\frac{1}{m}} = \frac{\binom{n}{2}}{m} = \frac{n(n-1)/2}{m} \leq \frac{n^2}{2m} \leq \frac{1}{2}$$

Then by Markov's inequality $\Pr[Y \geq 1] \leq \frac{1}{2}$ and so $\Pr[Y = 0] \geq \frac{1}{2}$.

Now we prove (ii) and assume that $n \geq c\sqrt{m}$. We could easily modify the argument above to show that $\mathbb{E}[Y]$ is large. But that by itself does not suffice⁷ to obtain a good enough lower bound on $\Pr[Y \geq 1]$. So we need to use more properties of Y . Note that it is not true that the random variables $Y_{i,j}$ are independent because $Y_{i,j} = 1 = Y_{j,k}$ implies already that $Y_{i,k} = 1$. But it is true that those random variables are *pairwise* independent. To see this, note that if i_1, j_1, i_2, j_2 are distinct indices then clearly $Y_{i_1, j_1}, Y_{i_2, j_2}$ are independent. So consider random variables $Y_{i,j}, Y_{j,k}$ with distinct i, j, k . Then for any outcomes $x_i, x_j \in \{1, \dots, m\}$ one has $\Pr[Y_{jk} \mid X_i = x_i, X_j = x_j] = \frac{1}{m}$ and so Y_{ij}, Y_{jk} must be independent. Then using Lemma 1.15 we can bound

$$\text{Var}[Y] = \sum_{1 \leq i < j \leq n} \text{Var}[Y_{ij}] = \binom{n}{2} \cdot \frac{1}{m} \cdot \left(1 - \frac{1}{m}\right) \leq \frac{\binom{n}{2}}{m}$$

Here we also use the insight from Theorem 1.16 that an indicator random variable has variance of $\text{Var}[Y_{ij}] = \Pr[Y_{ij} = 1] \cdot (1 - \Pr[Y_{ij} = 1])$. Then

$$\Pr[Y = 0] \leq \Pr[|Y - \mathbb{E}[Y]| \geq \mathbb{E}[Y]] \stackrel{\text{Chebychev}}{\leq} \frac{\text{Var}[Y]}{\mathbb{E}[Y]^2} = \frac{\binom{n}{2}/m}{(\binom{n}{2}/m)^2} = \frac{m}{\binom{n}{2}} = \frac{m}{n(n-1)/2} \leq \frac{2}{c^2} + o(1)$$

□

1.3.7 Hoeffding Inequality

We have seen how to use Chebychev's inequality in a setting where we need to control a sum of pairwise independent indicator random variables. But let us consider the following setting: we have independent random "coins" $X_1, \dots, X_n \in \{0, 1\}$ with $\Pr[X_i = 1] = \frac{1}{2} = \Pr[X_i = 0]$ for all $i = 1, \dots, n$. Let $X := \sum_{i=1}^n X_i$ be the number of coins that come up "head". Then one can easily see that $\text{Var}[X_i] = \frac{1}{4}$ and so $\text{Var}[X] = \frac{n}{4}$. Hence using Chebychev we can derive that the chance that all coins come up heads is

$$\Pr[X = n] \leq \Pr\left[|X - \underbrace{\mathbb{E}[X]}_{=n/2}| \geq \frac{n}{2}\right] \leq \frac{\text{Var}[X]}{(n/2)^2} = \frac{1}{n}$$

⁷If we have an integral random variable Y with $0 \leq Y \leq M$, then one can always infer that $\Pr[Y \geq 1] \geq \frac{\mathbb{E}[Y]}{M}$ but that would be too weak for the claim we are proving here.

But clearly we know that the probability of all coins being heads is 2^{-n} . So Chebyshev's inequality gives us a *polynomially* small probability where the actual probability is *exponentially* small. One can remedy this using a family of inequalities that have names like Chernov, Hoeffding, Bernstein or Azuma. While they slightly differ in their assumptions or statement, the mechanics of their proofs is almost identical. We will formulate two variants and prove one of them:

Theorem 1.18 (Hoeffding Inequality). *Let X_1, \dots, X_n be independent random variables so that $a_i \leq X_i \leq b_i$ for all $i \in [n]$ and let $X = \frac{1}{n}(X_1 + \dots + X_n)$. Then for any $\varepsilon > 0$,*

$$\Pr[|X - \mathbb{E}[X]| \geq \varepsilon] \leq 2 \exp\left(-\frac{2n^2\varepsilon^2}{\sum_{i=1}^n (b_i - a_i)^2}\right)$$

In the example of coin tosses, $X \in [0, 1]$ would be the fraction of the n coins that come up heads. Then setting $\varepsilon = \frac{1}{2}$ we can upper bound the probability that all coins come up heads by $\Pr[|X - \mathbb{E}[X]| \geq \frac{1}{2}] \leq 2 \exp(-\frac{2n^2(1/2)^2}{n}) = 2 \exp(-n/2)$ which is acceptably close to the actual probability of 2^{-n} for this event. For a vector $x \in \mathbb{R}^n$ we write its coordinates as $x = (x_1, \dots, x_n)$ and $\|x\|_2 = \sqrt{\sum_{i=1}^n x_i^2}$ is its *Euclidean norm*.

Theorem 1.19 (Azuma's Inequality). *Let X_1, \dots, X_n be independent random variables with $\mathbb{E}[X_i] = 0$ and $|X_i| \leq b_i$ for all i and let $X := \sum_{i=1}^n X_i$. Then for any $\lambda \geq 0$ one has*

$$\Pr[|X| \geq \lambda \|b\|_2] \leq 2 \exp(-\lambda^2/2)$$

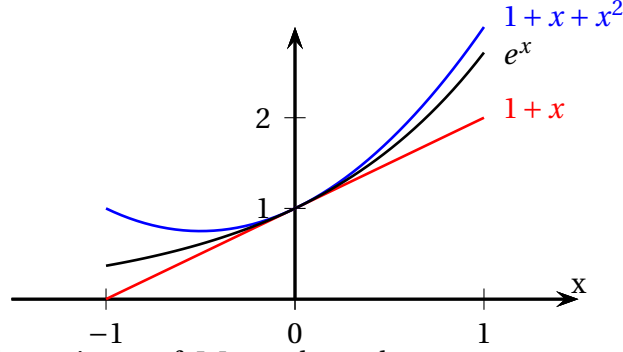
Proof. We prove a slightly weaker constant of $1/4$ instead of $1/2$. By symmetry and the union bound it suffices to show that $\Pr[X \geq \lambda \|b\|_2] \leq \exp(-\lambda^2/4)$. Let $t > 0$ be a parameter that we decide later. The proof strategy is to upper bound the *exponential moment* $\mathbb{E}[\exp(tX)]$. The intuition is that if this exponential moment is small, then events where X is large should be exponentially unlikely.

Claim I. *For each i one has $\mathbb{E}[\exp(tX_i)] \leq \exp(t^2 b_i^2)$.*

Proof of Claim I. If $t \cdot b_i > 1$ then $\mathbb{E}[\exp(tX_i)] \leq \exp(tb_i) \leq \exp(t^2 b_i^2)$ by monotonicity of the exponential function and we are done. So suppose that $t \cdot b_i \leq 1$. Then

$$\mathbb{E}[\exp(tX_i)] \stackrel{(I)}{\leq} \mathbb{E}[1 + tX_i + (tX_i)^2] = 1 + \underbrace{t\mathbb{E}[X_i]}_{=0} + \underbrace{t^2\mathbb{E}[X_i^2]}_{\leq b_i^2} \leq 1 + t^2 b_i^2 \stackrel{(II)}{\leq} \exp(t^2 b_i^2)$$

Here we use in (I) that $e^x \leq 1 + x + x^2$ for $-1 \leq x \leq 1$ and in (II) we use the fact that $1 + x \leq e^x$ for all $x \in \mathbb{R}$. \square



Now back to the main proof. We can bound

$$\mathbb{E}[\exp(tX)] = \mathbb{E}\left[\prod_{i=1}^n \exp(tX_i)\right] \stackrel{(III)}{=} \prod_{i=1}^n \mathbb{E}[\exp(tX_i)] \stackrel{\text{Claim 1}}{\leq} \prod_{i=1}^n \exp(t^2 b_i^2) = \exp(t^2 \|b\|_2^2)$$

Here we crucially use in (III) that for *independent* random variables Y and Z one has $\mathbb{E}[YZ] = \mathbb{E}[Y]\mathbb{E}[Z]$. Then using Markov's inequality on the exponential moment we obtain

$$\Pr[X > \lambda \|b\|_2] \stackrel{(IV)}{=} \Pr[e^{tX} > e^{t\lambda \|b\|_2}] \stackrel{\text{Markov}}{\leq} \frac{\mathbb{E}[e^{tX}]}{e^{t\lambda \|b\|_2}} \leq e^{t^2 \|b\|_2^2 - t\lambda \|b\|_2} \stackrel{(V)}{=} e^{-\lambda^2/4}$$

Here we use the monotonicity of $x \mapsto e^{tx}$ in (IV). Finally in (V) we have minimized the quadratic function $t \mapsto t^2 \|b\|_2^2 - t\lambda \|b\|_2$ by choosing $t := \frac{\lambda}{2\|b\|_2}$. \square

For convenience, we also state a version of Hoeffding's inequality where we do not consider X as the *average* but the *sum* of the individual random variables.

Theorem 1.20 (Hoeffding Inequality II). *Let X_1, \dots, X_n be independent random variables so that $a_i \leq X_i \leq b_i$ for all $i \in [n]$ and let $X = X_1 + \dots + X_n$. Then for any $t > 0$,*

$$\Pr[|X - \mathbb{E}[X]| \geq t] \leq 2 \exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right)$$

1.3.8 Polling

Let us revisit the polling application from earlier. We can tighten the result and show that the *empirical mean* $Y := \frac{1}{n} \sum_{i=1}^n X_i$ is much more likely to be near its expectation μ than we first proved in Theorem 1.16. By some abuse of notation we write $\text{Var}[\mathcal{D}]$ as the variance of a random variable that is drawn from \mathcal{D} .

Theorem 1.21. *Let \mathcal{D} be a distribution over $\{0, 1\}$. Take independent random samples $X_1, \dots, X_n \sim \mathcal{D}$ and let $Y := \frac{1}{n} \sum_{i=1}^n X_i$ be the empirical mean. Then for any $\varepsilon > 0$, $\Pr[|Y - \mathbb{E}[Y]| \geq \varepsilon] \leq 2 \exp(-2n\varepsilon^2)$.*

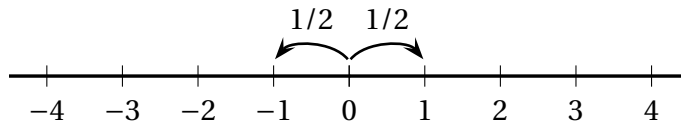
Proof. Use Hoeffding's Inequality with $a_i = 0$, $b_i = 1$. \square

For example this means that it takes $n = \frac{\ln(2/\delta)}{2\varepsilon^2}$ many samples so that the chance that Y deviates from the expectation by more than ε is at most δ .

1.3.9 Random walk on a line

A *random walk* is random process that produces a path where do not explicitly describe the path but only the individual (random) steps. A random walk often is a walk in a graph or in Euclidean space \mathbb{R}^d . Here we describe a simple random walk on the line.

The process starts at time 0 at position 0. In step i we do one step to the right with probability $1/2$ and one step to the left with probability $1/2$, independently of previous steps.



Let $X_i \in \{-1, 1\}$ be a random variable with $\Pr[X_i = 1] = \frac{1}{2} = \Pr[X_i = -1]$, then $X := \sum_{i=1}^n X_i$ models the position after n iterations. Then $\text{Var}[X_i] = 1$ for each i and so $\text{Var}[X] = n$. Then the standard deviation is $\sqrt{\text{Var}[X]} = \mathbb{E}[X^2]^{1/2} = \sqrt{n}$ meaning that on average after n steps we expect to be roughly \sqrt{n} steps away from the origin. And we can prove that being much further away is unlikely:

Lemma 1.22. For any $\lambda \geq 0$ one has $\Pr[|X| \geq \lambda\sqrt{n}] \leq 2e^{-\lambda^2/4}$.

Proof. Simply apply Azuma's Inequality with step size $b_i = 1$. \square

1.4 Discrepancy theory

For a vector $y \in \mathbb{R}^n$ we define

$$\|y\|_\infty := \max\{|y_i| : i \in [n]\}$$

as the *maximums norm* or ℓ_∞ -norm and

$$\|y\|_1 := \sum_{i=1}^n |y_i|$$

is the ℓ_1 -norm.

Consider a matrix $A \in \{0, 1\}^{m \times n}$. Our goal is to find a vector $x \in \{-1, 1\}^n$ so that $\|Ax\|_\infty$ is as small as possible. Often this is phrased as an equivalent set problem where A is the incidence matrix of a set system with n elements and m sets and the goal is to color the elements with $+1$'s and -1 's so that each set is approximately evenly colored. While explicitly picking the colors and getting low discrepancy is not easy, we will prove that a *random* choice of a coloring is a good option.

Lemma 1.23. *Let $A \in \{0, 1\}^{m \times n}$ and draw $x \sim \{-1, 1\}^n$ uniformly at random. Then*

$$\Pr[\|Ax\|_\infty \leq 2\sqrt{\ln(4m)} \cdot \sqrt{n}] \geq \frac{1}{2}.$$

Proof. Let $\lambda > 0$ be a parameter that we decide later. Consider a row $i \in [m]$. We note that the inner product $\langle A_i, x \rangle = \sum_{j:A_{ij}=1} x_j$ is the sum of $\|A_i\|_1$ many independent ± 1 random variables and so, following Lemma 1.22 we have

$$\Pr[|\langle A_i, x \rangle| \geq \lambda\sqrt{n}] \stackrel{n \geq \|A_i\|_1}{\leq} \Pr[|\langle A_i, x \rangle| \geq \lambda\sqrt{\|A_i\|_1}] \leq 2e^{-\lambda^2/4}$$

The discrepancy for different rows are not independent but we can use the union bound (Lemma 1.10) to get

$$\Pr[\exists i \in [m] : |\langle A_i, x \rangle| \geq \lambda\sqrt{n}] \leq \sum_{i=1}^m \Pr[|\langle A_i, x \rangle| \geq \lambda\sqrt{n}] \leq 2m \cdot e^{-\lambda^2/4} = \frac{1}{2}$$

where the last step follows from choosing $\lambda := 2\sqrt{\ln(4m)}$. □

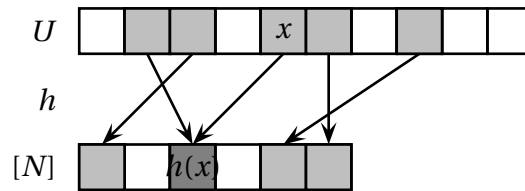
We note that for $m = n$ one can prove that there is always a vector $x \in \{-1, 1\}^n$ with $\|Ax\|_\infty \leq 6\sqrt{n}$. However, this is not a uniform randomly chosen one, see Spencer [Spe85]. But the argument above can be slightly tweaked to give a coloring of discrepancy $2\sqrt{\ln(4m) \cdot k}$ where $k := \max_i \|A_i\|_1$ is the maximum number of ones per row.

1.5 Hashing

Suppose our goal is to build a *data structure* to store a set E of entries. For example these entries could be files, each having a unique file name and we want to store the files using an array. Let U be the *universe* (e.g. the set of all possible file names) so that $E \subseteq U$ and let N be the predetermined length of the array. For some function $h : U \rightarrow \{1, \dots, N\}$ we will store entry $x \in U$ at position $h(x)$. The

function h is called a *hash function* and should not depend on the set E (for example E might be unknown before the creation of the data structure or it might be dynamically changing over time).

It can happen that there are several entries $x_1, x_2 \in E$ with $h(x_1) = h(x_2)$ — we call this a *collision*. We can store all entries $h^{-1}(k) \cap E$ at position k in a linearly ordered list but that means accessing any of those entries will cost us time proportional to the number of collision $|h^{-1}(k) \cap E|$. Ideally one would hope to construct a hash function h so that even for $|E| \approx N$ the average lookup-time for each element is $O(1)$.



The reader may note that a deterministic construction of h is not going to be good enough — whatever h we choose, U is in general gigantic and we could be unlucky that the entries are precisely of the form $E \subseteq h^{-1}(k)$ for some k (meaning they would all be stored in the same position).

The natural solution is to choose the hash function h at *random* from a family \mathcal{H} of hash functions. To warm up we state two observations concerning uniform random hash functions:

Lemma 1.24. *Let \mathcal{H} be the set of all functions of the form $h : U \rightarrow [N]$. Fix entries $E \subseteq U$ with $|E| \leq \sqrt{N}$ and draw $h \sim \mathcal{H}$ at random. Then with probability at least $1/2$, the restriction $h|_E : E \rightarrow [N]$ is injective (i.e. there are no collisions).*

In other words, if $|E| \leq \sqrt{N}$ then with good probability we do not have any collisions. This is exactly what we have shown for the birthday paradox, see Theorem 1.17. But it seems very inefficient to reserve space $|E|^2$ to store $|E|$ many elements. For the case $|E| = N$ we can prove the following:

Lemma 1.25. *Let \mathcal{H} be the set of all functions $h : U \rightarrow [N]$. Then for any fixed $E \subseteq U$ with $|E| = N$ and $h \sim \mathcal{H}$ the following holds with probability at least $1 - \frac{1}{N}$:*

- (i) *The average look-up time of an entry is $\mathbb{E}_{x \sim E}[|h^{-1}(h(x)) \cap E|] \leq O(1)$.*
- (ii) *The maximum number of entries mapped to one position is $\max_{k \in [N]} |h^{-1}(k) \cap E| \leq O(\log N / \log \log N)$.*

We will leave these statements as an exercise to prove. So even though there may be some positions k where a super constant number of entries are being

stored, the average access time is $O(1)$ which would be sufficient for us. But storing a random function $h : U \rightarrow [N]$ would require $\Theta(|U| \log(N))$ many bits which is prohibitive as $|U| \gg N$. Then the next thought is that maybe we can limit the randomness needed to construct the hash function.

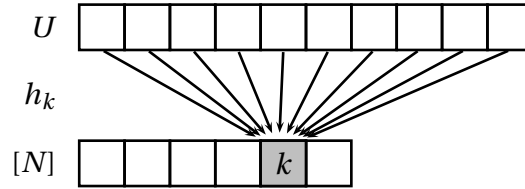
1.6 Limited independence

If we construct a family of hash functions \mathcal{H} then one property seems particularly natural: for any fixed $x \in U$, as one draws a function $h \sim \mathcal{H}$ from the family, we want that the index $h(x)$ is uniformly distributed over $[N]$.

Definition 1.26. Let \mathcal{H} be a family of hash functions from U to $[N]$. We say that \mathcal{H} is *one-way independent* if

$$\Pr_{h \sim \mathcal{H}} [h(x) = k] = \frac{1}{N} \quad \forall x \in U \quad \forall k \in [N]$$

But this property is not enough for a useful family of hash functions. For example let $h_k : U \rightarrow [N]$ be the function with $h_k(x) := k$ for all $x \in U$.



Then the family of hash functions $\mathcal{H} := \{h_1, \dots, h_N\}$ is indeed one-way independent but even maximizes the number of possible collisions. So we need a stronger property:

Definition 1.27. Let \mathcal{H} be a family of hash functions from U to $[N]$. We say that \mathcal{H} is *pair-wise independent* if for all distinct $x_1, x_2 \in U$ and all $k_1, k_2 \in [N]$ one has

$$\Pr_{h \sim \mathcal{H}} [h(x_1) = k_1 \text{ and } h(x_2) = k_2] = \frac{1}{N^2}$$

Intuitively this means that only considering two entries x_1, x_2 , $h \sim \mathcal{H}$ looks like the uniform distribution. Later we will see that we do not need this probability to be exactly $\frac{1}{N^2}$; it suffices to have an upper bound up to some small factor α .

Definition 1.28. Let \mathcal{H} be a family of hash functions from U to $[N]$. We say that \mathcal{H} is α -approximately pair-wise independent if for all distinct $x_1, x_2 \in U$ and all $k_1, k_2 \in [N]$ one has

$$\Pr_{h \sim \mathcal{H}} [h(x_1) = k_1 \text{ and } h(x_2) = k_2] \leq \frac{\alpha}{N^2}$$

We will later explain how $O(1)$ -approximately pair-wise independent families can be constructed using limited randomness. But first we explain how to use them!

1.6.1 The birthday paradox revisited

We revisit the result from Section 1.3.6 and discuss the power of the approximate pair-wise independence.

Theorem 1.29. Let \mathcal{H} be a family of hash functions from $h : U \rightarrow [N]$ that are α -approximately pair-wise independent. Then for any $E \subseteq U$ with $\alpha \cdot |E|^2 \leq N$ one has

$$\Pr_{h \sim \mathcal{H}} [\text{no collision in } h|_E] \geq \frac{1}{2}$$

Proof. We write $E = \{x_1, \dots, x_n\}$. We will adjust the analysis from the original birthday paradox. For indices $1 \leq i < j \leq n$, let $Y_{ij} \in \{0, 1\}$ denote the indicator random variable telling whether $h(x_i) = h(x_j)$, meaning we have a collision for the pair. We can estimate that

$$\mathbb{E}[Y_{ij}] = \Pr_{h \sim \mathcal{H}} [h(x_i) = h(x_j)] = \sum_{k \in [N]} \underbrace{\Pr[h(x_i) = k = h(x_j)]}_{\leq \frac{\alpha}{N^2}} \leq N \cdot \frac{\alpha}{N^2} = \frac{\alpha}{N}$$

Now, let

$$Y := \sum_{1 \leq i < j \leq n} Y_{ij}$$

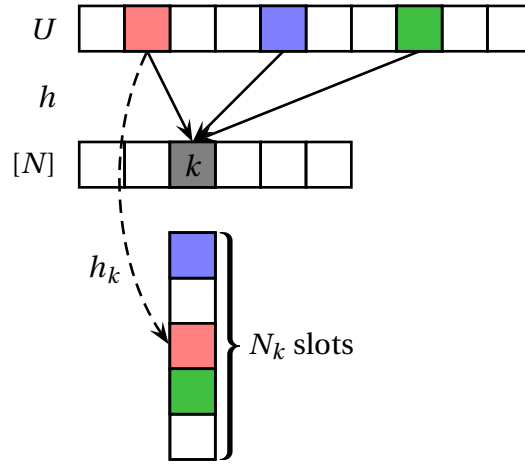
be the total number of colliding pairs. Then

$$\mathbb{E}[Y] = \sum_{1 \leq i < j \leq n} \mathbb{E}[Y_{ij}] \leq \binom{n}{2} \cdot \frac{\alpha}{N} \leq \frac{n^2}{2N} \leq \frac{1}{2}$$

Then by Markov's inequality $\Pr[Y \geq 1] \leq \frac{1}{2}$ and so the probability of no collision is $\Pr[Y = 0] \geq \frac{1}{2}$. \square

1.6.2 Double Hashing

We will now discuss a result of Fredman, Komlós and Szemerédi [FKS84] which uses two levels of hash functions to create a data structure with lookup time of $O(1)$ and space requirement $O(|E|)$. Given elements $E \subseteq U$ with $n := |E|$ we set $N := n$ and let \mathcal{H} be a family of α -approximately pairwise independent hash functions. Draw a hash function $h \sim \mathcal{H}$. For $k \in [N]$, let $Z_k := |h^{-1}(k) \cap E|$ be the random variable that denotes the number of elements mapped to k . For each $k \in [N]$, let \mathcal{H}_k be another family of α -approximate hash functions, this time from U to $[N_k]$ where $N_k := \alpha Z_k^2$. We draw $h_k \sim \mathcal{H}_k$; if there is any collision for $h_k|_{h^{-1}(k) \cap E}$ then we repeat the draw of h_k .



Note that by Theorem 1.29 the probability that there is no collision is at least $1/2$ and it can be verified in time $O(N_k)$.

If we want to look-up the position of an element $x \in E$, then we first check position $k := h(x)$. Then we check position $h_k(x)$ in the sub-array located at k . The total look-up time is clearly $O(1)$ for every element $x \in E$.

Theorem 1.30. *The expected space requirement of double hashing is $O(\alpha^2 n)$.*

Proof. Again, we write $E = \{x_1, \dots, x_n\}$. First note that the number of elements mapped to position k is

$$Z_k = \sum_{i=1}^n \mathbf{1}_{h(x_i)=k}$$

Squaring and taking expectations gives

$$\begin{aligned} \mathbb{E}[Z_k^2] &= \mathbb{E}_{h \sim \mathcal{H}} \left[\left(\sum_{i=1}^n \mathbf{1}_{h(x_i)=k} \right)^2 \right] \\ &= \mathbb{E}_{h \sim \mathcal{H}} \left[\underbrace{\sum_{i=1}^n \mathbf{1}_{h(x_i)=k}}_{=Z_k} \right] + \sum_{1 \leq i < j \leq n} \underbrace{\mathbb{E}[\mathbf{1}_{h(x_i)=k=h(x_j)}]}_{\leq \alpha/N^2} \leq \mathbb{E}[Z_k] + \frac{\alpha \binom{n}{2}}{N^2} \stackrel{N=n}{\leq} \mathbb{E}[Z_k] + \frac{\alpha}{2} \end{aligned}$$

where we use the property of α -approximately independent hash functions. Then

$$\mathbb{E} \left[\sum_{k=1}^N \alpha Z_k^2 \right] \leq \alpha \cdot N \cdot \frac{\alpha}{2} + \alpha \underbrace{\sum_{k=1}^N \mathbb{E}[Z_k]}_{=n} \leq O(\alpha^2 n)$$

□

1.6.3 Construction of pairwise independent hash functions

After we have seen the usefulness of the concept we also want to now show how to actually construction pairwise independent hash functions. We fix a prime number p . For $a, b \in \{0, \dots, p-1\}$ we define the function $f_{a,b} : \{0, \dots, p-1\} \rightarrow \{0, \dots, p-1\}$ with

$$f_{a,b}(x) := ax + b \pmod{p}$$

We can prove that picking the parameters a, b at random gives a pair-wise independent hash function.

Lemma 1.31. *Let $x, y, s, t \in \{0, \dots, p-1\}$ with $x \neq y$. Then*

$$\Pr_{a,b} [f_{a,b}(x) = s \text{ and } f_{a,b}(y) = t] = \frac{1}{p^2}$$

where we draw $a, b \sim \{0, \dots, p-1\}$ uniformly at random.

Proof. We can rewrite

$$\begin{bmatrix} f_{a,b}(x) = s \\ f_{a,b}(y) = t \end{bmatrix} \Leftrightarrow \begin{bmatrix} ax + b \equiv_p s \\ ay + b \equiv_p t \end{bmatrix} \Leftrightarrow \begin{bmatrix} a(x-y) \equiv_p s-t \\ b \equiv_p s-ax \end{bmatrix}$$

Now let us consider the probability that these two equations are satisfied as we draw $a, b \sim \{0, \dots, p-1\}$. Since $x-y \not\equiv_p 0$ and p is a prime, there is a unique a that makes $a(x-y) \equiv_p s-t$ true, hence this happens with probability $\frac{1}{p}$. Then once a is fixed there is a unique b that makes $b \equiv_p s-ax$ true. Together this gives the claim. □

This is *almost* what we need to get 2-wise independent hash functions for our setting. There is only the minor issue that we need a function $h : U \rightarrow [N]$ instead of $f : \{0, \dots, p-1\} \rightarrow \{0, \dots, p-1\}$. For that purpose we fix a prime p with⁸ $|U| \leq p \leq 2|U|$. Then we take everything modulo N instead of p while suffering a small constant factor error.

Lemma 1.32. *For $|U| \geq 4N$, let p be a prime number with $|U| \leq p \leq 2|U|$ and write $U = \{0, \dots, |U| - 1\}$. Let $\mathcal{H} := \{h_{a,b} : a, b \in \{0, \dots, p-1\}\}$ be the family of functions $h_{a,b} : U \rightarrow \{0, \dots, N-1\}$ with $h_{a,b}(x) := ax + b \pmod{N}$. Then \mathcal{H} is 2-approximately pairwise independent.*

Proof. Let $x, y \in U$ and $k \in \{0, \dots, N-1\}$. Then

$$\begin{aligned} \Pr_{a,b} [h_{a,b}(x) = h_{a,b}(y) = k] &= \sum_{s,t \in \{0, \dots, p-1\}, s \equiv_N t \equiv_N k} \underbrace{\Pr[f_{a,b}(x) = s \text{ and } f_{a,b}(y) = t]}_{= \frac{1}{p^2}} \\ &= \frac{1}{p^2} \cdot |\{s \in \{0, \dots, p-1\} : s \equiv_N k\}|^2 \leq \frac{1}{p^2} \left\lceil \frac{p}{N} \right\rceil^2 \leq \frac{2}{N^2} \end{aligned}$$

□

We would like to point out that by taking $p \gg N$ we can make the hash function $(1 + \varepsilon)$ -approximately pairwise independent for any desired $\varepsilon > 0$. So in applications we can for the sake of the analysis pretty much assume to have exactly pairwise independent functions as we can make any error as small as needed.

This concludes the construction of an (approximate) pairwise independent function. One might wonder whether one could generalize the construction to independence of larger tuples. And indeed this is possible.

Definition 1.33. A family \mathcal{H} of functions from U to $[N]$ is called *k-wise independent* if for all distinct $x_1, \dots, x_k \in U$ and all $a_1, \dots, a_k \in [N]$ one has

$$\Pr_{h \sim \mathcal{H}} [h(x_1) = a_1, \dots, h(x_k) = a_k] = \frac{1}{N^k}$$

Now, fix a prime number p and for $a_0, \dots, a_{k-1} \in \{0, \dots, p-1\}$ we define a function $f_{a_0, \dots, a_{k-1}} : \{0, \dots, p-1\} \rightarrow \{0, \dots, p-1\}$ with

$$f_{a_0, \dots, a_{k-1}}(x) := \sum_{i=0}^{k-1} a_i x^i \pmod{p}$$

⁸This is actually a very nontrivial statement. But the Bertrand-Chebyshev Theorem proven in 1852 says that indeed for any $n \geq 2$ there is a prime number p with $n < p < 2n$. In fact for large enough n , any interval $[n, n + n^{0.525}]$ must contain a prime number and the (unproven) Riemann hypothesis would imply that intervals of the form $[n, n + n^{0.5+o(1)}]$ contain primes.

In other words, this is a univariate polynomial of degree $k - 1$. We take all these polynomials to define a hash family

$$\mathcal{H} := \{f_{a_0, \dots, a_{k-1}} : a_0, \dots, a_{k-1} \in \{0, \dots, p-1\}\}$$

Then one can prove the following:

Theorem 1.34. \mathcal{H} is k -wise independent.

We will not prove this formally but we sketch the argument.

Sketch. Fix distinct x_1, \dots, x_k and fix any $b_1, \dots, b_k \in \{0, \dots, p-1\}$. Then

$$\begin{aligned} \begin{bmatrix} f_{a_0, \dots, a_{k-1}}(x_1) & = & b_1 \\ & \vdots & \\ f_{a_0, \dots, a_{k-1}}(x_k) & = & b_k \end{bmatrix} &\Leftrightarrow \left[\sum_{i=0}^{k-1} a_i x_j^i \equiv_p b_j \forall j = 1, \dots, k \right] \\ &\Leftrightarrow \underbrace{\begin{pmatrix} 1 & 1 & \dots & 1 \\ x_1 & x_2 & \dots & x_k \\ x_1^2 & x_2^2 & \dots & x_k^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{k-1} & x_2^{k-1} & \dots & x_k^{k-1} \end{pmatrix}}_{=:X} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{k-1} \end{pmatrix} \equiv_p \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_k \end{pmatrix} \end{aligned}$$

The matrix X is also called a *Vandermonde matrix*. One can prove that for distinct x_1, \dots, x_k , the matrix X is *invertible* (over \mathbb{F}_p). That implies that there is exactly one solution $a = (a_0, \dots, a_{k-1})$ to the linear system and the probability to draw exactly that solution is $\frac{1}{p^k}$. \square

1.7 Unbiased estimators

We want to extend the argument from Section 1.3.5. We say that a random variable X is an *unbiased estimator* for a number $\mu > 0$ if $\mathbb{E}[X] = \mu$. We are interested in the question how many samples from X are needed to determine μ up to some error ε . In Section 1.3.5 we considered the *absolute error* i.e. we wanted to find an s with $|\mu - s| \leq \varepsilon$. Now we will ask for a *relative error* which means finding a number s with $(1 - \varepsilon)\mu \leq s \leq (1 + \varepsilon)\mu$ which typically is more useful in particular if the number μ is small. The following quantity will be useful for that purpose:

Definition 1.35. For a random variable X with $\mathbb{E}[X] > 0$ we define the *relative variance* as

$$t(X) := \frac{\text{Var}[X]}{\mathbb{E}[X]^2}$$

Theorem 1.36. *Let X be an unbiased estimator of $\mu > 0$ and let $\varepsilon > 0$. Then by taking $k := O(\frac{t(X)}{\varepsilon^2})$ samples from X we can determine μ up to a $1 \pm \varepsilon$ factor with probability at least $\frac{9}{10}$.*

Proof. For $k := \frac{10t(X)}{\varepsilon^2}$, let X_1, \dots, X_k be independent samples of X and let $Y := \frac{1}{k}(X_1 + \dots + X_k)$ be their average. Then $\mathbb{E}[Y] = \mathbb{E}[X] = \mu$ and using Chebychev's inequality (Theorem 1.14) we have

$$\Pr[|Y - \mu| > \varepsilon\mu] \leq \frac{\text{Var}[Y]}{(\varepsilon\mu)^2} = \frac{1}{\varepsilon^2 k} \frac{\text{Var}[X]}{\mu^2} = \frac{t(X)}{\varepsilon^2 k} \leq \frac{1}{10}$$

□

This argument uses only Chebychev's inequality. But the samples are independent and one might be optimistic that using the more powerful inequality of Hoeffding (Theorem 1.18) we could prove a lower error probability than a constant such as $\frac{1}{10}$. For example we would like to prove that $k = \Theta(\log(1/\delta) \frac{t(X)}{\varepsilon^2})$ samples suffices so that $\Pr[|Y - \mu| \leq \varepsilon\mu] \geq 1 - \delta$. But Theorems 1.18 and 1.19 require a fixed interval that contains any outcome of the random samples. And indeed this is needed as X might have *heavy tails*.

Let us consider the following example. We draw $Z \sim (0, 1]$ uniformly at random and set $X := \frac{1}{Z^{1/4}}$. Then $\mu := \mathbb{E}[X] = \int_0^1 \frac{1}{x^{1/4}} dx = \frac{4}{3}$ and

$$\text{Var}[X] \leq \mathbb{E}[X^2] = \int_0^1 \frac{1}{x^{1/2}} dx = 2$$

Now if Y is the average of k samples of X , then

$$\Pr[|Y - \mu| \geq \varepsilon\mu] \geq \Pr\left[Z \leq \left(\frac{1}{k} \cdot \frac{4}{3} \varepsilon\right)^4\right] = \Theta\left(\frac{\varepsilon^4}{k^4}\right)$$

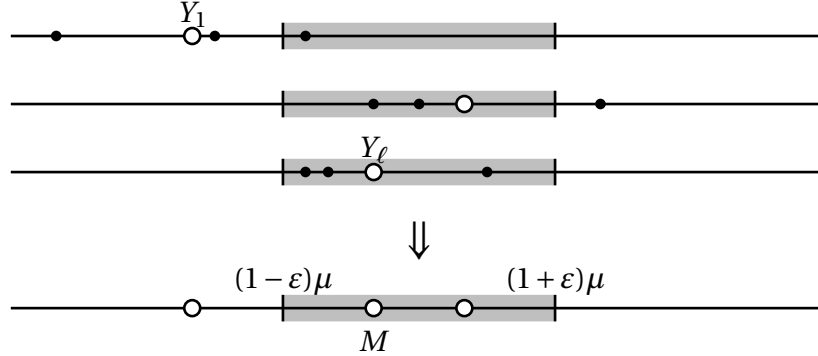
because a single outlier sample might already ruin the average. The trick is to take ℓ -batches of k -samples each. From each batch we take the average, then we take the median of those averages.

Theorem 1.37. *Let $\varepsilon, \mu > 0$, $0 < \delta < 1$ and let X be an unbiased estimator of μ . Then with $\Theta(\frac{t(X)}{\varepsilon^2} \log \frac{1}{\delta})$ many independent samples of X one can construct a number Z so that $\Pr[|Z - \mu| \leq \varepsilon\mu] \geq 1 - \delta$.*

Proof. We assume $\log(\frac{1}{\delta}) \geq 1$. Let $k := \frac{10t(X)}{\varepsilon^2}$. Recall that if Y is the average of k samples from X , then by Theorem 1.36 we know that

$$\Pr[(1 - \varepsilon)\mu \leq Y \leq (1 + \varepsilon)\mu] \geq \frac{9}{10}$$

Now, for a parameter $\ell \in \mathbb{N}$ that we determine later, we draw ℓ many independent samples of Y that we call Y_1, \dots, Y_ℓ . Then let M denote the *median* of Y_1, \dots, Y_ℓ .



Claim. One has $\Pr[|M - \mu| \geq \epsilon\mu] \leq e^{-\ell/8}$.

Proof of Claim. For $i = 1, \dots, \ell$ we define the indicator random variable

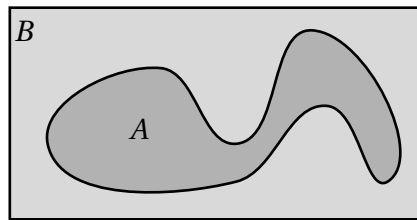
$$Z_i := \begin{cases} 1 & \text{if } (1 - \epsilon)\mu \leq Y_i \leq (1 + \epsilon)\mu \\ 0 & \text{otherwise} \end{cases}$$

Then we know that $\Pr[Z_i = 1] \geq \frac{9}{10}$ for all i . If the median M does not lie in the interval $[(1 - \epsilon)\mu, (1 + \epsilon)\mu]$ then at least half the Y_i 's need to lie outside of it and so $\sum_{i=1}^{\ell} Z_i \leq \frac{\ell}{2}$. But Hoeffding's Inequality we have

$$\Pr \left[\sum_{i=1}^{\ell} Z_i \leq \frac{\ell}{2} \right] \leq \Pr \left[\left| \sum_{i=1}^{\ell} Z_i - \mathbb{E} \left[\sum_{i=1}^{\ell} Z_i \right] \right| \geq \frac{\ell}{4} \right] \leq e^{-\ell/8} \quad \square$$

Then choosing $\ell := 8 \log(1/\delta)$ gives the claim. □

We want to describe one application. Suppose we have a set $A \subseteq \mathbb{R}^d$ and we would like to compute its volume that we denote by $\text{Vol}_d(A)$. But we assume that the only access to A that we have is that for any given point $x \in \mathbb{R}^d$ we can test whether $x \in A$ or not. Additionally let us assume that there is a set B with $A \subseteq B \subseteq \mathbb{R}^d$ so that we can efficiently sample a point from B .



The we can obtain the following:

Theorem 1.38 (Monte Carlo method). *Let $A \subseteq B \subseteq \mathbb{R}^d$. From $k := O\left(\frac{\text{Vol}_d(B)}{\text{Vol}_d(A)} \cdot \frac{\log(1/\delta)}{\varepsilon^2}\right)$ many uniform samples from B one can determine $\text{Vol}_d(A)$ up to a $1 \pm \varepsilon$ factor with probability $1 - \delta$.*

Proof. Let p be a uniform random point from B and let X be the indicator random variable

$$\begin{cases} 1 & \text{if } p \in A \\ 0 & \text{otherwise} \end{cases}$$

Next, define $Y := \text{Vol}_d(B) \cdot X$. Then $\mathbb{E}[Y] = \text{Vol}_d(B) \cdot \mathbb{E}[X] = \text{Vol}_d(B) \cdot \frac{\text{Vol}_d(A)}{\text{Vol}_d(B)} = \text{Vol}_d(A)$. Hence Y is an unbiased estimator of $\text{Vol}_d(A)$. Using that relative variance is scaling-invariant⁹ we have

$$t(Y) = t(X) = \frac{\text{Var}[X]}{\mathbb{E}[X]^2} \leq \frac{\mathbb{E}[X^2]}{\mathbb{E}[X]^2} = \frac{\text{Vol}_d(A)/\text{Vol}_d(B)}{(\text{Vol}_d(A)/\text{Vol}_d(B))^2} = \frac{\text{Vol}_d(B)}{\text{Vol}_d(A)}$$

Then the claim follows from Theorem 1.37. □

1.8 Streaming

We will see a non-trivial application of the unbiased estimator result. Suppose we have a streaming setting where we see a stream of data but the amount of data is so huge that we cannot hold it all in our memory and we also can see the stream only once. The goal in *streaming algorithms* is to still be able to answer certain questions about the data stream while using very limited memory.

More concretely, suppose $U = \{1, \dots, |U|\}$ is a huge universe of numbers and the data stream consists of numbers $x_1, \dots, x_n \in U$ that we see one after the other. For $i \in U$, let

$$f_i := \# \text{ of times that } i \text{ appears in } x_1, \dots, x_n$$

denote the *frequency* of element i . We also define

$$F_k := \sum_{i=1}^{|U|} f_i^k$$

where we make the convention of $0^0 = 0$. Then F_k is the k th *moment* of the frequency vector $f = (f_1, \dots, f_{|U|})$. In particular

- F_0 is the number of distinct elements in the sequence.

⁹I.e. for any random variable $X > 0$ and any constant $s > 0$ one has $t(sX) = \frac{\text{Var}[sX]}{\mathbb{E}[sX]^2} = \frac{s^2 \text{Var}[X]}{s^2 \mathbb{E}[X]^2} = t(X)$.

- One as $F_1 = n$.
- F_2 is the second moment vector of f .

There is a trivial algorithm to compute each F_k : we have a counter for each number $f_i \in U$ and simply count what f_i is. But that takes $O(|U| \log(n))$ space which is prohibitive. We will instead prove the following remarkable result

Theorem 1.39 (Alon, Matias, Szegedy [AMS96]). *There is a streaming algorithm for a sequence $x_1, \dots, x_n \in U$ that with space $O(\frac{\log(n|U|)}{\epsilon^2} \cdot \log(\frac{1}{\delta}))$ computes a $1 \pm \epsilon$ approximation to F_0 and F_2 with probability $1 - \delta$.*

We leave the case of F_0 for the homework and prove how to approximately compute F_2 .

Here is an observation that is going to be the basis of our streaming algorithm:

Lemma 1.40. *Let $f \in \mathbb{R}^U$. For all $i \in U$, draw $h(i) \sim \{-1, 1\}$ independently and uniformly at random for all $i \in U$ and set $Y := \sum_{i \in U} f_i h(i)$. Then $\mathbb{E}[Y^2] = \sum_{i \in U} f_i^2$.*

That means Y^2 is an unbiased estimator for the quantity F_2 . Moreover Y can be computed by simply adding up $h(x_j)$ in each step. However there is one issue which is that drawing a random function h and keeping it in our memory would require $O(|U|)$ space which again is more than we have. So the next trick we use is that it suffices to have a function h which is $O(1)$ -wise independent. Combining Theorem 1.34 and Lemma 1.32 with $N = 2$ immediately gives that for any fixed k one can construct a family \mathcal{H} of functions $h : U \rightarrow \{-1, 1\}$ that is $(1 + \epsilon')$ -approximately k -wise independent. Here one can make the error ϵ' so small that it is irrelevant (in fact it can even be of the order $\epsilon' = \frac{1}{\text{poly}(|U|, n)}$). Another subtle point is that the family \mathcal{H} constructed via Theorem 1.34 is not just k -wise independent for a fixed k , but it is 1-wise, 2-wise, 3-wise, \dots , k -wise independent. We will now return to our streaming application and justify that being $O(1)$ -wise independent is enough for us:

Lemma 1.41. *Fix any $f \in \mathbb{R}^U$ and let \mathcal{H} be a family of function $h : U \rightarrow \{-1, 1\}$ that is 1-wise, 2-wise and 4-wise independent. Then the random variable $Y := \sum_{i \in U} f_i h(i)$ satisfies:*

- One has $\mathbb{E}[Y^2] = \|f\|_2^2$
- One has $\text{Var}[Y^2] \leq 2\|f\|_2^4$

Proof. First we prove (i). For distinct $i, j \in U$ one has $\mathbb{E}[h(i)h(j)] = 0$. Hence

$$\mathbb{E}[Y^2] = \mathbb{E}\left[\left(\sum_{i \in U} f_i h(i)\right)^2\right] = \sum_{i \in U} f_i^2 \underbrace{\mathbb{E}[h(i)^2]}_{=1} + \sum_{i, j \in U: i \neq j} f_i f_j \underbrace{\mathbb{E}[h(i)h(j)]}_{=0} = \|f\|_2^2$$

Next we show (ii). We have

$$\begin{aligned} \mathbb{E}[Y^4] &= \mathbb{E}\left[\left(\sum_{i \in U} f_i h(i)\right)^4\right] \\ &= \sum_{i_1, i_2, i_3, i_4 \in U} f_{i_1} f_{i_2} f_{i_3} f_{i_4} \mathbb{E}[h(i_1)h(i_2)h(i_3)h(i_4)] \\ &\stackrel{(*)}{=} \binom{4}{2} \sum_{i, j \in U: i < j} f_i^2 f_j^2 + \sum_{i \in U} f_i^4 \end{aligned}$$

where it remains to justify (*). Since h is 4-wise independent, one has $\mathbb{E}[h(i_1)h(i_2)h(i_3)h(i_4)] = \mathbb{E}[g(i_1)g(i_2)g(i_3)g(i_4)]$, where $g : U \rightarrow \{-1, 1\}$ is a (truly) uniform random function. If there is any index, say i_1 , that appears with multiplicity 1 in i_1, i_2, i_3, i_4 , then we can pull it out and $\mathbb{E}[g(i_1)g(i_2)g(i_3)g(i_4)] = \mathbb{E}[g(i_1)] \cdot \mathbb{E}[g(i_2)g(i_3)g(i_4)] = 0$. So the only non-zero terms must come from the cases that 2 indices appears twice and from the case that $i_1 = i_2 = i_3 = i_4$. For any ordered pair of indices $i < j$ there are $\binom{4}{2} = 6$ possible permutations. We continue the estimate on the variance:

$$\begin{aligned} \text{Var}[Y^2] &= \mathbb{E}[(Y^2)^2] - \mathbb{E}[Y^2]^2 \\ &= 6 \sum_{i, j \in U: i < j} f_i^2 f_j^2 + \sum_{i \in U} f_i^4 - \left(\sum_{i \in U} f_i^2\right)^2 = 4 \sum_{i, j \in U: i < j} f_i^2 f_j^2 \leq 2 \left(\sum_{i \in U} f_i^2\right)^2 \end{aligned}$$

□

For an error of say $\delta = \frac{1}{10}$ the streaming algorithm that estimates F_2 is as follows:

<p>SINGLE ESTIMATE STREAMING ALGORITHM TO APPROXIMATE F_2 Input: Parameter $\varepsilon > 0$. Data stream $x_1, \dots, x_n \in U$ Output: Estimate on F_2 with error ε and error probability $\frac{1}{10}$ (1) For $k := \Theta(\frac{1}{\varepsilon^2})$, independently draw functions $h_1, \dots, h_k : U \rightarrow \{-1, 1\}$ so that each individual h_i is 1-wise, 2-wise, 4-wise independent. (2) Set $Y_1 := Y_2 := \dots := Y_k := 0$ (3) For $i = 1$ to n (4) Read x_i from the stream (5) FOR $j = 1$ to k DO update $Y_j := Y_j + h_j(x_i)$ (6) Return the average of Y_1^2, \dots, Y_k^2</p>
--

Then the error probability can be brought down by taking the median of independent runs:

STREAMING ALGORITHM TO APPROXIMATE F_2

Input: Parameters $\varepsilon, \delta > 0$. Data stream $x_1, \dots, x_n \in U$

Output: Estimate on F_2 with error ε and error probability δ .

- (1) Run SINGLE ESTIMATE independently in parallel $s := 8 \log(1/\delta)$ times.
Let $Z^{(1)}, \dots, Z^{(s)}$ be the returned values
- (2) Return the median value in $Z^{(1)}, \dots, Z^{(s)}$.

The correctness follows from the earlier discussion. Now, let us discuss the memory requirement. Each counter takes on values in $\{-n|U|, \dots, n|U|\}$ and needs $O(\log(n|U|))$ bits. Each hash function needs $O(\log|U|)$ bits. We need to keep $sk = \Theta(\frac{1}{\varepsilon^2} \log \frac{1}{\delta})$ counters and hash functions in memory. That gives the claim.

Chapter 2

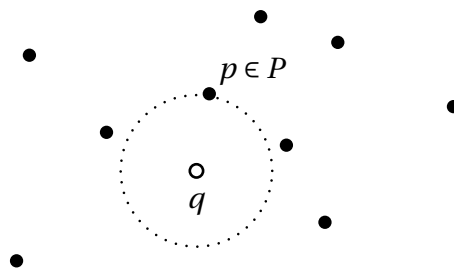
The curse of dimensionality and dimension reduction

In this chapter, we want to discuss geometric problems and algorithms.

2.1 The nearest neighbor problem

The *nearest neighbor problem* is the following: We are given a set of points $P \subseteq \mathbb{R}^d$ with $n := |P|$. We are allowed to do some preprocessing and construct a data structure so that we can answer the following query: for an incoming query point $q \in \mathbb{R}^d$, find a point $p \in P$ that attains

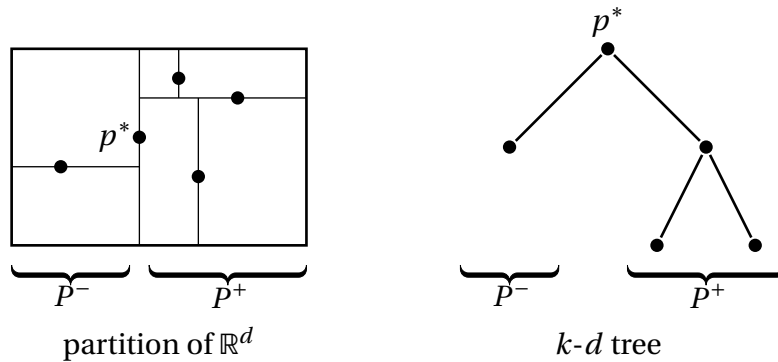
$$\min_{p \in P} \text{dist}(p, q)$$



Example for $d = 2$ and dist being Euclidean distance

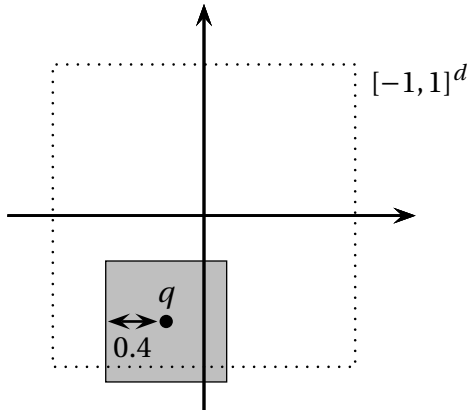
Here depending on the application, the distance metric dist might be the Euclidean distance, the maximums norm $\| \cdot \|_{\infty}$ or the Manhattan norm $\| \cdot \|_1$. One can imagine that this problem may appear as a frequent subproblem in applications where one has to deal with large point sets.

A popular data structure to such a maintain geometric point sets is a k - d tree. This data structure is constructed as follows: we select a coordinate $i \in [d]$ and a point $p^* \in P$. Then we split the remaining points into $P \setminus \{p^*\} = P^- \cup P^+$ where $P^- := \{p \in P \setminus \{p^*\} \mid p_i \leq p_i^*\}$ and $P^+ = \{p \in P \setminus \{p^*\} \mid p_i > p_i^*\}$. Then we recurse on P^- and P^+ . In terms of how to select that point p^* and the coordinate, a good rule seems to be to cycle through the coordinates $i = 1, \dots, d$ and pick a split point p so that p_i is the median of the coordinates p_i . This naturally induces a tree with root p^* and it also induces a partition of \mathbb{R}^d into *cells*.



Assuming we use the median splitting point, the depth of the tree will be $O(\log n)$. Then given a query point q we can find the cell containing q in time $O(\log n)$ by going down the tree, starting at the root. There is also a heuristic algorithm to use a k - d tree to find a nearest neighbor. The idea is that given a query point q , if we have already found a point p' , then any cell whose minimum distance to q is at least $\text{dist}(q, p')$ can be pruned. To be the best of my knowledge, there is no good worst case analysis for the query time for such an algorithm but there are those papers that assume that the query point q is drawn *random* from some distribution and even assuming randomness the query times scale proportional to 2^d .

We want to give an intuitive argument where this exponential dependence might come from. Suppose that $P \subseteq \mathbb{R}^d$ with $|P| \leq 2^d$ and our k - d tree has led to the partition of \mathbb{R}^d into the 2^d many open cells corresponding to the quadrants (i.e. their boundaries are the hyperplanes $x_i = 0$ for $i = 1, \dots, d$). Now draw a random point $q \sim [-1, 1]^d$ and suppose we want to use the k - d tree partition to decide whether P contains a point at $\|\cdot\|_\infty$ -distance at most 0.4. First note that for every fixed $p \in P$ one has $\Pr_q[\|p - q\|_\infty \leq 0.4] \leq 0.4^d$ and so very likely there is no point in P at distance at most 0.4 to q . On the other hand consider the indices $I := \{i \in [d] : |q_i| \leq 0.4\}$. Very likely one has say $|I| \geq \Omega(d)$. Then $q + 0.4 \cdot [-1, 1]^d$ intersects $2^{|I|} = 2^{\Theta(d)}$ of the cells.

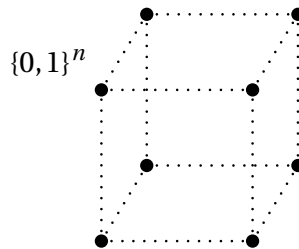


That means any algorithm based on the k - d tree needs to search in $2^{\Theta(d)}$ many cells for a nearest point in P . Similar running time blow ups appear in many geometry based problems. This is typically referred to as the *curse of dimensionality*.

In the following, we want to describe a clever approach for our problem that circumvents this curse of dimensionality. First we relax the problem and only require to find an *approximately* closest point. We also make a concrete choice for the distance function and restrict the points to be from $\{0, 1\}^d$. This will simplify notation and we will visit extensions in the homework.

Approximate Nearest Neighbor Search ANNS(c, r). Given a set of points $P \subseteq \{0, 1\}^d$, the distance metric $\text{dist}(x, y) := \|x - y\|_1$ and a parameter $r > 0$. Build a data structure so that for an incoming query point $q \in \{0, 1\}^d$ with $\text{dist}(q, P) \leq r$ we can produce a point $p \in P$ with $\text{dist}(p, q) \leq c \cdot r$.

Here $c > 1$ is the error that we allow ourselves. Note that for points $x, y \in \{0, 1\}^d$, the quantity $\|x - y\|_1$ simply tells the number of coordinates in which x and y differ.



2.1.1 Locally sensitive hash functions

The idea is to use a *locally sensitive hash function*, which means a hash function h where the value $h(p)$ depends on the coordinates of the point p .

Lemma 2.1. *There is a family \mathcal{H} of functions of the form $h : \{0, 1\}^d \rightarrow \{0, 1\}$ so that for all $p, q \in \{0, 1\}^d$:*

$$\Pr_{h \sim \mathcal{H}} [h(p) = h(q)] = 1 - \frac{\|p - q\|_1}{d}$$

Proof. Let $h_i : \{0, 1\}^d \rightarrow \{0, 1\}$ be the function with $h_i(p) := p_i$, i.e. the function returns the i th coordinate. Set $\mathcal{H} := \{h_1, \dots, h_d\}$. Then

$$\Pr_{h \sim \mathcal{H}} [h(p) = h(q)] = \frac{|\{i \in [d] : p_i = q_i\}|}{d} = 1 - \frac{\|p - q\|_1}{d}$$

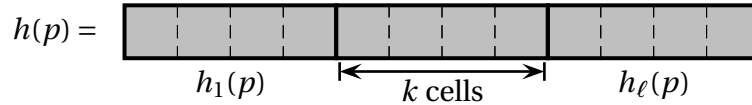
□

However, for points p, q , the cases $\|p - q\|_1 \leq r$ vs $\|p - q\|_1 \geq cr$ lead to at most a constant factor gap in the probability of being mapped to the same value (and that only if $r = \Theta(d)$). This would not be enough to be helpful for our setting. The next trick is to *amplify* this gap!

Let \mathcal{H} be the family of d hash functions from Lemma 2.1. For parameters $k, \ell \in \mathbb{N}$ that we decide later, we draw $k \cdot \ell$ many hash function $h_{i,j} \sim \mathcal{H}$ for $i \in [\ell]$, $j \in [k]$ and we write

$$h(p) := (h_{i,j}(p))_{i \in [\ell], j \in [k]} = \underbrace{(h_1(p), \dots, h_\ell(p))}_{\in \{0,1\}^k} \tag{2.1}$$

We think of each $h_i(p) \in \{0, 1\}^k$ as a *block*, i.e. we have ℓ blocks.



Using this combined hash function h will increase the previous gap drastically:

Theorem 2.2. *Let $p, q \in \{0, 1\}^d$ and let $cr \leq d$ and $c \geq 8$. Let h be a random hash function as in (2.1) for suitable choices of k and ℓ . Then*

(i) *If $\|p, q\|_1 \leq r$ then $\Pr_h[\exists \text{ block } i : h_i(p) = h_i(q)] \geq 1 - \frac{1}{n^2}$*

(ii) *If $\|p - q\|_1 \geq cr$ then $\Pr_h[\exists \text{ block } i : h_i(p) = h_i(q)] \leq \frac{1}{n^2}$*

Proof. For a fixed block $i \in [\ell]$ the probability to have a collision is

$$\begin{aligned}
\Pr[h_i(p) = h_i(q)] &= \prod_{j \in [k]} \Pr[h_{ij}(p) = h_{ij}(q)] \\
&\stackrel{\text{Lem 2.1}}{=} \left(1 - \frac{\|p - q\|_1}{d}\right)^k \\
&\begin{cases} \geq \left(1 - \frac{r}{d}\right)^k & \text{if } \|p - q\|_1 \leq r \\ \leq \left(1 - \frac{cr}{d}\right)^k & \text{if } \|p - q\|_1 \geq cr \end{cases} \\
&(*) \begin{cases} \geq \exp(-2kr/d) & \text{if } \|p - q\|_1 \leq r \\ \leq \exp(-kcr/d) & \text{if } \|p - q\|_1 \geq cr \end{cases} \\
&\stackrel{k := \frac{4d \ln(n)}{cr}}{=} \begin{cases} \frac{1}{n^{8/c}} & \text{if } \|p - q\|_1 \leq r \\ \frac{1}{n^4} & \text{if } \|p - q\|_1 \geq cr \end{cases}
\end{aligned}$$

In (*) we use that $1 - x \leq e^{-x}$ for all $x \in \mathbb{R}$ for the upper bound. For the lower bound we use $1 - x \geq e^{-2x}$ for $0 \leq x \leq \frac{1}{2}$ and the fact that $\frac{r}{d} \leq \frac{1}{2}$. The reader may note that we have increased the *constant gap* from Lemma 2.1 to a polynomial gap. Unfortunately now both bounds are $\ll 1$ so we need to increase the collision probability by having many blocks.

For this purpose we set $\ell := 2 \ln(n) \cdot n^{8/c}$. For (i) we have

$$\Pr[\exists i \in [\ell] : h_i(p) = h_i(q)] \geq 1 - \left(1 - \frac{1}{n^{8/c}}\right)^\ell \geq 1 - \exp\left(-\frac{\ell}{n^{8/c}}\right) = 1 - \frac{1}{n^2}$$

using again $1 - x \leq e^{-x}$ for all x meaning that we have boosting the collision probability to close to 1. For (ii) we get $\Pr[\exists i \in [\ell] : h_i(p) = h_i(q)] \leq \ell \cdot \frac{1}{n^4} \leq \frac{1}{n^2}$ using the union bound. \square

Here the assumption of $c \geq 8$ was arbitrary; one can make the argument work with any $c > 1$ at the expense of the other parameters.

LSH ALGORITHM

Input: Points $P \subseteq \{0, 1\}^d$, parameters $r > 0$ and $c > 2$.

Preprocessing:

- (1) Draw hash function h (consisting of $k \cdot \ell$ individual functions)
- (2) For all i , sort $\{h_i(p) : p \in P\}$ (interpreting these as numbers)

Query: $q \in \{0, 1\}^d$

- (1) Compute $h(q)$
- (2) FOR all i , use binary search to find a point $p \in P$ with $h_i(p) = h_i(q)$.
Return the first that was found.

Theorem 2.3. *The data structure answers queries correctly with high probability; it uses memory $\tilde{O}(n^{1+8/c})$ and the query time is $\tilde{O}(n^{2/c})$ where \tilde{O} hides polylogarithmic terms.*

Proof. For correctness, suppose that there is a point $p^* \in P$ with $\|p^* - q\|_1 \leq r$. Then by Theorem 2.2.(i) with high probability there is a block i so that $h_i(p^*) = h_i(q)$. On the other hand, by Theorem 2.2.(ii) and the union bound over n points in P , there is no $p \in P$ with $\|p - q\|_1 \geq cr$ so that for some i one would have $h_i(p) = h_i(q)$. That means the only colliding points have distance $\leq cr$ and the algorithm returns one of those.

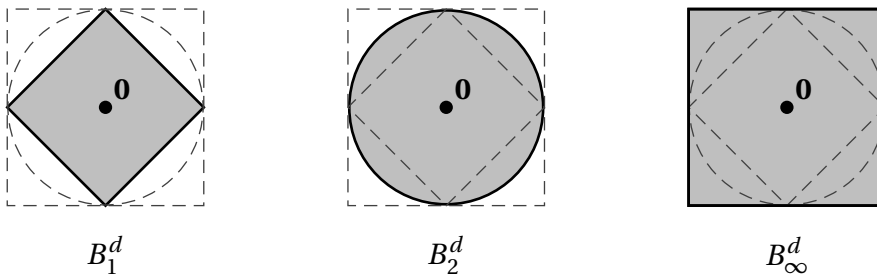
Next, we note that the data structure has size $\ell \cdot k \cdot \Theta(n) = O(d^2 \ln(n) n^{1+8/c})$ which is the amount to store the hash function values for the n points. For a query we need to do ℓ binary searches with vectors of bit length k , so the query time is $O(\log^3(n) \cdot d \cdot n^{8/c})$. □

2.2 Volume in higher dimensions

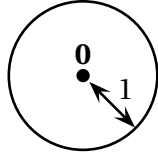
We want to define a few useful high dimension objects. For a point $c \in \mathbb{R}^d$ (the *center*) and a radius $r > 0$ we define:

- The *Euclidean ball* $B_2^d(c, r) := \{x \in \mathbb{R}^d \mid \|x - c\|_2 \leq r\}$
- The ℓ_∞ ball (or *cube*) $B_\infty^d(c, r) := \{x \in \mathbb{R}^d \mid \|x - c\|_\infty \leq r\}$
- The ℓ_1 ball is $B_1^d(c, r) := \{x \in \mathbb{R}^d \mid \|x - c\|_1 \leq r\}$.

We also abbreviate $B_2^d := B_2^d(\mathbf{0}, 1)$, $B_\infty^d := B_\infty^d(\mathbf{0}, 1)$ and $B_1^d := B_1^d(\mathbf{0}, 1)$ as the corresponding balls of radius 1 centered at the origin. For $d = 2$ the picture is as follows:



We also define $S^{d-1} := \{x \in \mathbb{R}^d \mid \|x\|_2 = 1\}$ as the *sphere* in \mathbb{R}^d . Note that even though this object lives in \mathbb{R}^d it is $(d - 1)$ -dimensional and so for historic reasons it is common to use $d - 1$ as index.



S^{d-1} for $d = 2$

We define the *volume* or *Lebesgue measure* of a set $D \subseteq \mathbb{R}^d$ as

$$\text{Vol}_d(D) := \int_D 1 \, dx$$

In dimension $d = 2$ this corresponds to the area. We note that for any center c and any radius r one has

$$B_1^d(c, r) \subseteq B_2^d(c, r) \subseteq B_\infty^d(c, r)$$

By looking at a 2-dimensional or 3-dimensional picture one would get the impression that while the cube $B_\infty^d(c, r)$ is larger than the Euclidean ball $B_2^d(c, r)$, it does not seem to be much larger. But this intuition is wrong in higher dimensions. We will fill in the details:

Lemma 2.4. *In dimension d , one has $\frac{\text{Vol}_d(B_2^d)}{\text{Vol}_d(B_\infty^d)} \leq e^{-\Theta(d)}$.*

Proof. We use a *Monte Carlo* argument: we draw a uniform point $x \sim B_\infty^d$ (which means we draw $x_1, \dots, x_d \sim [-1, 1]$ independently) and upper bound the probability of landing in B_2^d , i.e. we need to prove that¹ $\Pr[\sum_{i=1}^d x_i^2 \leq 1]$ is tiny. Let $X_i := x_i^2$. Then $0 \leq X_i \leq 1$ for any outcome and $\mathbb{E}[X_i] = \int_{-1}^1 t^2 dt = \frac{1}{3} t^3 \Big|_{t=-1}^1 = \frac{2}{3}$. Hence $X := X_1 + \dots + X_d$ has $\mathbb{E}[X] = \frac{2}{3}d$. Then by Hoeffding Inequality II (Theorem 1.20) one has

$$\begin{aligned} \Pr[x \in B_2^d] &= \Pr[X \leq 1] \leq \Pr\left[|X - \mathbb{E}[X]| \geq \frac{d}{3} - 1\right] \\ &\stackrel{\text{Thm 1.20}}{\leq} 2 \exp\left(-2 \cdot \frac{(\frac{d}{3} - 1)^2}{d \cdot (1 - 0)}\right) \stackrel{d \geq 5}{\leq} 2 \exp\left(-2 \cdot \frac{d}{18}\right) \end{aligned}$$

□

This bound is not actually tight. Clearly $\text{Vol}_d(B_\infty^d) = 2^d$. Computing the volume of the Euclidean ball takes much more care, but one can prove that for even d one has $\text{Vol}_d(B_2^d) = \frac{\pi^{d/2}}{(d/2)!}$. From this one can derive that

$$\frac{\text{Vol}_d(B_2^d)}{\text{Vol}_d(B_\infty^d)} \leq d^{-\Theta(d)}$$

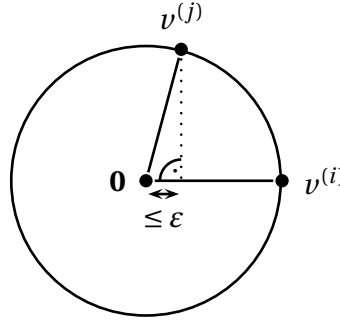
¹Here it will be useful to drop the $\sqrt{\cdot}$.

For the interested reader we recommend the excellent survey by Ball [Bal97] which covers volumes in higher dimensions in detail.

2.3 Nearly orthogonal vectors

We want to discuss a different phenomenon where the high dimensional case behaves very differently from what one might expect from a 2-dimensional figure. We could ask how many distinct non-zero vectors can one have in \mathbb{R}^d that are pairwise orthogonal? The answer is (as one might expect) d and it is attained for example for the standard basis e_1, \dots, e_d . Now we relax the problem and ask how many vectors $v^{(1)}, \dots, v^{(m)} \in \mathbb{R}^d$ can one have that are *nearly* orthogonal, say the pairwise angles are between 89° and 91° . Even though the answer seems to be the same for say $d = 2$ and $d = 3$, in higher dimension d the answer is $2^{\Theta(d)}$.

Theorem 2.5. *For any $\varepsilon > 0$, there are $v^{(1)}, \dots, v^{(m)} \in S^{d-1}$ with $m \geq e^{\Theta(d/\varepsilon^2)}$ and $|\langle v^{(i)}, v^{(j)} \rangle| \leq \varepsilon$ for all $i \neq j$.*



Proof. We choose the vectors $v^{(1)}, \dots, v^{(m)}$ independently and uniformly from $\{-\frac{1}{\sqrt{d}}, +\frac{1}{\sqrt{d}}\}^d$. Note that indeed $\|v^{(i)}\|_2 = 1$ for all $i = 1, \dots, m$. We need to prove that very likely, the vectors are pairwise orthogonal. It suffices to analyze a single pair: **Claim.** Fix $u \in \{-\frac{1}{\sqrt{d}}, +\frac{1}{\sqrt{d}}\}^d$. Draw $v \sim \{-\frac{1}{\sqrt{d}}, \frac{1}{\sqrt{d}}\}^d$. Then $\Pr[|\langle u, v \rangle| \geq \varepsilon] \leq 2e^{-2d\varepsilon^2}$.

Proof of Claim. Set $X_k := u_k v_k$. Then we can write $X := \langle u, v \rangle = \sum_{k=1}^d u_k v_k = \sum_{k=1}^d X_k$. Note that X_1, \dots, X_d are independent and each X_k attains the values $-\frac{1}{d}$ and $\frac{1}{d}$ with equal probability. In particular $\mathbb{E}[X_k] = 0$. Then using again Hoeffding Inequality II, we have

$$\Pr[|X| \geq \varepsilon] \leq 2 \exp\left(-2 \frac{\varepsilon^2}{d \cdot (1/d)^2}\right) = 2 \exp(-2d\varepsilon^2) \quad \square$$

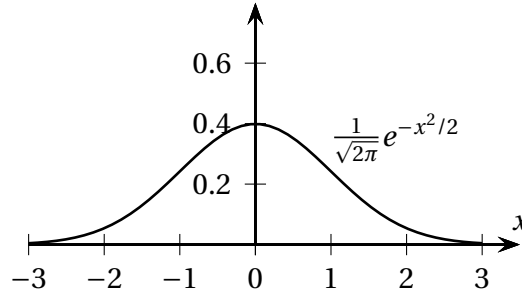
Now back to the main argument. Let $p := 2 \exp(-2d\varepsilon^2)$. Choose m so that $m^2 p \leq \frac{1}{2}$. Then by the union bound,

$$\Pr[\forall i \neq j : |\langle v^{(i)}, v^{(j)} \rangle| \leq \varepsilon] = 1 - \Pr[\exists i \neq j : |\langle v^{(i)}, v^{(j)} \rangle| > \varepsilon] \geq 1 - \binom{m}{2} \cdot p \geq \frac{1}{2}$$

□

2.4 Introductions to Gaussians

The *1-dimensional (standard) Gaussian distribution* $\mathcal{N}(0, 1)$ is the distribution on \mathbb{R} that has the *density function* $\frac{1}{\sqrt{2\pi}} e^{-x^2/2}$.



Gaussian density

A d -dimensional Gaussian random vector $g \in \mathbb{R}^d$ is a vector where all coordinates are drawn independently from $\mathcal{N}(0, 1)$. Note that the density function of such a Gaussian random vector must be the product of the density functions of its coordinates². Hence the density at $x \in \mathbb{R}^d$ is

$$\prod_{i=1}^d \frac{1}{\sqrt{2\pi}} e^{-x_i^2/2} = \frac{1}{(2\pi)^{d/2}} e^{-\|x\|_2^2/2} \quad (2.2)$$

We make a definition:

Definition 2.6. A distribution \mathcal{D} over \mathbb{R}^d is called *rotationally invariant* if the distribution of $\langle u, v \rangle$ with $u \sim \mathcal{D}$ is the same for all $v \in S^{d-1}$.

We have seen in (2.2) that the Gaussian density at a point x only depends on $\|x\|_2$ and not on the direction of x which should mean that the distribution is rotationally invariant. We will verify that this intuition is indeed correct. First, recall that the sum of independent Gaussians is again Gaussian.

²This holds more generally for any *product distribution*, i.e. any vector whose coordinates are independently distributed.

Lemma 2.7. Let $\alpha, \beta \in \mathbb{R}$ and let $g_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $g_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$ be two independent Gaussians. Then $\alpha g_1 + \beta g_2$ has the same distribution as $\mathcal{N}(\alpha\mu_1 + \beta\mu_2, \alpha^2\sigma_1^2 + \beta\sigma_2^2)$.

One can prove this by inspecting the density function of the sum, which we skip here.

Lemma 2.8. The Gaussian (standard) distribution is rotationally invariant. In particular for any $v \in S^{d-1}$ and a Gaussian random vector $g \in \mathbb{R}^d$ one has $\langle g, v \rangle \sim \mathcal{N}(0, 1)$.

Proof. Recall that $g = (g_1, \dots, g_d)$ where the coordinates have been independently drawn from $\mathcal{N}(0, 1)$. Then

$$\langle v, g \rangle = \sum_{i=1}^d v_i g_i \stackrel{\text{Lem 2.7}}{\sim} \mathcal{N}(0, v_1^2 + \dots + v_d^2) = \mathcal{N}(0, 1)$$

□

We also would like to mention that one can define a (non-standard) Gaussian with mean $\mu \in \mathbb{R}^d$ and a positive definite³ covariance matrix $\Sigma \in \mathbb{R}^{d \times d}$ that has the density

$$\frac{1}{\sqrt{\det(2\pi\Sigma)}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$

If $X = (X_1, \dots, X_d)$ is a random vector with that density then $\mathbb{E}[X] = \mu$ and $\mathbb{E}[XX^T] = \Sigma$. It is worth noting that for $\mu = \mathbf{0}$ and $\Sigma = I_d$ we recover the standard Gaussian. We also would like to mention that whenever Σ is not a scalar of the identity matrix I_d , then the (non-standard) Gaussian is not rotationally invariant.

2.5 More on rotationally invariant distributions

We can also give an alternative characterization of rotationally invariant distributions.

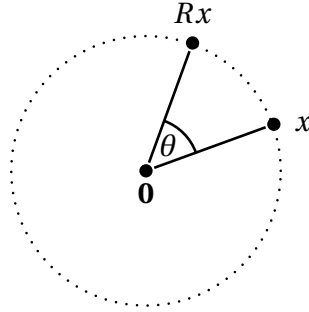
Definition 2.9. A matrix $R \in \mathbb{R}^{n \times n}$ is called a *rotation matrix* if $\|Rx\|_2 = \|x\|_2$ for all $x \in \mathbb{R}^n$.

In dimension $n = 2$ the matrix

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

³We will define and discuss this term later.

can be seen to be a rotation matrix. Note that R is the matrix that *rotates* a point x by degree θ (hence the name rotation matrix).



One can prove that every rotation matrix in $n = 2$ is of this form. Note that the columns of R have Euclidean length $(\cos(\theta)^2 + \sin(\theta)^2)^{1/2} = 1$ and they are orthogonal. This observation can be generalized. We say that vectors $v_1, \dots, v_m \in \mathbb{R}^d$ are *orthonormal* if $\|v_i\|_2 = 1$ for all $i = 1, \dots, m$ and $\langle v_i, v_j \rangle = 0$ for all $i \neq j$.

Lemma 2.10. *Let $R \in \mathbb{R}^{n \times n}$. Then the following two conditions are equivalent:*

- (i) R is a rotation matrix.
- (ii) The columns R^1, \dots, R^n are orthonormal.

Proof. We first prove the easier direction, which is (ii) \Rightarrow (i). If the columns of R are orthonormal one can see that $R^T R = I_n$. Then for any $x \in \mathbb{R}^n$ one has $\|Rx\|_2^2 = (Rx)^T (Rx) = x^T R^T R x = x^T I_n x = \|x\|_2^2$. Now we show (i) \Rightarrow (ii). Since R is a rotation matrix we know that for all $i \in [n]$ one has $\|R^i\|_2^2 = \|R e_i\|_2^2 = \|e_i\|_2^2 = 1$. Next, for $i \neq j$ one has

$$\underbrace{\|R^i\|_2^2}_{=1} + \underbrace{\|R^j\|_2^2}_{=1} + 2 \langle R^i, R^j \rangle = \|R^i + R^j\|_2^2 = \|R(e_i + e_j)\|_2^2 = \|e_i + e_j\|_2^2 = 2$$

from which one derives that $\langle R^i, R^j \rangle = 0$.

□

2.6 Concentration of measure for Gaussians

Random Gaussians are often used in randomized algorithms. Other than rotation invariance, they have remarkable concentration properties. If we draw a random Gaussian $X = (X_1, \dots, X_d)$, i.e. $X_i \sim \mathcal{N}(0, 1)$, then $\mathbb{E}[\|X\|_2^2] = \sum_{i=1}^d \mathbb{E}[X_i^2] = d$. Deviating much from this expectation is unlikely:

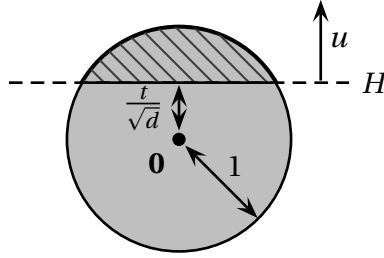
Theorem 2.11. Let $X = (X_1, \dots, X_d)$ be a standard Gaussian random vector and let $0 < \varepsilon < 1$. Then

$$\Pr \left[\left| \|X\|_2^2 - d \right| \geq \varepsilon d \right] \leq 2 \exp \left(-\frac{d\varepsilon^2}{8} \right)$$

We skip the proof. Note that it does not directly follow from Hoeffding's Inequality (Theorem 1.18) because there is no fixed value M with $|X_i| \leq M$. But one can modify the proof strategy of Theorem 1.19 and show that for each coordinate i and each $t > 0$ one has $\mathbb{E}[e^{tX_i}] = e^{t^2/2}$; the rest of the proof is then straightforward.

We will now use Theorem 2.11 to show that in dimension d , 99% of the volume of B_2^d lie along a narrow belt of width $O(\frac{1}{\sqrt{d}})$.

Theorem 2.12. Let $u \in \mathbb{R}^d$ be a vector with $\|u\|_2 = 1$ and let $t \geq 1$. The fraction of the volume of B_2^d that lies above the hyperplane $H = \{x \in \mathbb{R}^d \mid \langle x, u \rangle \geq \frac{t}{\sqrt{d}}\}$ is at most $e^{-\Theta(t^2)}$.



Proof. It suffices to prove the claim for a point drawn from the *surface* of B_2^d rather than the interior (since stretching a point until it lies on the boundary may only increase the absolute value of the inner product with u). Since the Gaussian is rotationally invariant, $\frac{g}{\|g\|_2}$ is a uniform random point⁴ from S^{d-1} . By rotational invariance we may also assume that $u = e_1$. Then

$$\Pr \left[\left\langle u, \frac{g}{\|g\|_2} \right\rangle \geq \frac{t}{\sqrt{d}} \right] = \Pr \left[g_1 \geq \frac{t}{\sqrt{d}} \|g\|_2 \right] \leq \Pr \left[g_1^2 \geq t^2 \frac{\|g\|_2^2}{d} \right] =: (*)$$

The latter expression is somewhat inconvenient as it contains the two random variables g_1^2 and $\|g\|_2^2$ on both sides. But then we already know from Theorem 2.11 that $\|g\|_2^2$ is very unlikely to be as large as say $\frac{d}{2}$. Hence we can use the union bound to get

$$(*) \leq \Pr \left[g_1^2 \geq \frac{t^2}{2} \right] + \Pr \left[\|g\|_2^2 \leq \frac{d}{2} \right] \stackrel{t \geq 1}{\leq} e^{-\Theta(t^2)} + e^{-\Theta(d)}$$

⁴Well one might argue that this is not well defined if $\|g\|_2 = 0$. But as the Gaussian is a continuous distribution and hence the probability of $g = \mathbf{0}$ is equal to 0.

Here to bound the first term, we use the following fact that can be derived from the density function of the Gaussian.

Fact. For any $s > 0$ and $X \sim \mathcal{N}(0, 1)$ one has $\Pr[X \geq s] \leq \frac{1}{\sqrt{2\pi}s} e^{-s^2/2}$.

The claim then follows whenever $t \leq \sqrt{d}$. However, if $t > \sqrt{d}$ then the probability of the event in question is 0 since B_2^d does not contain any point of length $\frac{t}{\sqrt{d}} > 1$. That finishes the argument. \square

2.7 Dimension reduction

Finally we want to give a positive result that is often used to design faster algorithms when dealing with high dimensional data. Suppose we have a set of points $x_1, \dots, x_n \in \mathbb{R}^d$ that we need to do certain computations with. Typically any single “atomic” computation involving vectors costs running time $\Theta(d)$. So it would be wonderful if we could *reduce the dimension* of the vectors. Any embedding into any lower dimensional space will incur some error. But if the only quantity that needs to be preserved are Euclidean distances $\|x_i - x_j\|_2$ up to some $1 \pm \varepsilon$ error then one can reduce the dimension to $O(\frac{\log n}{\varepsilon^2})$.

Theorem 2.13 (Johnson Lindenstrauss Projection). *Let $0 < \varepsilon \leq 1$. For any points $x_1, \dots, x_n \in \mathbb{R}^d$ there is a linear map $T : \mathbb{R}^d \rightarrow \mathbb{R}^k$ with $k := \Theta(\frac{\log n}{\varepsilon^2})$ so that*

$$(1 - \varepsilon) \|x_i - x_j\|_2 \leq \|T(x_i) - T(x_j)\|_2 \leq (1 + \varepsilon) \|x_i - x_j\|_2 \quad \forall i, j \in [n]$$

Proof. We define our map T as $T(x) := Gx$ where $G \in \mathbb{R}^{k \times d}$ is a matrix where all entries are chosen independently as $G_{ij} \sim \mathcal{N}(0, \frac{1}{k})$.

It remains to prove that this choice of T works with high probability for an appropriate choice of k . First we want to understand how $T(y)$ behaves for a fixed vector $y \in \mathbb{R}^d$.

Claim I. *For fixed $y \in \mathbb{R}^d$, $T(y)$ is a random vector with independent coordinates where for each $i \in [k]$, $T(y)_i$ is distributed as $\mathcal{N}(0, \frac{\|y\|_2^2}{k})$.*

Proof of Claim I. First we note that the i th coordinate of $T(y)$ is

$$T(y)_i = \langle G_i, y \rangle = \sum_{j=1}^d G_{ij} y_j \sim \mathcal{N}\left(0, \frac{\|y\|_2^2}{k}\right)$$

by Lemma 2.7. Then the independence follows from the fact that the rows of G are drawn independently. \square

In particular for any $y \in \mathbb{R}^d$ one has $\mathbb{E}[\|T(y)\|_2^2] = \sum_{i=1}^k \mathbb{E}[T(y)_i^2] = k \cdot \frac{\|y\|_2^2}{k} = \|y\|_2^2$ which means that $T(y)$ has the correct expected length. Next, we prove that the

length also concentrates well around its expectation.

Claim I. For any $y \in \mathbb{R}^d$ one has $\Pr[|\|T(y)\|_2 - \|y\|_2| > \varepsilon \|y\|_2] \leq e^{-\varepsilon^2 k/8}$.

Proof of Claim. The claim is invariant under scaling y , hence we may assume that $\|y\|_2 = 1$. Then

$$\begin{aligned} \Pr[1 - \varepsilon \leq \|T(y)\|_2 \leq 1 + \varepsilon] &= \Pr[(1 - \varepsilon)^2 \leq \|T(y)\|_2^2 \leq (1 + \varepsilon)^2] \\ &\geq \Pr[1 - \varepsilon \leq \|T(y)\|_2^2 \leq 1 - \varepsilon] \stackrel{\text{Thm 2.11}}{\geq} 1 - 2 \exp\left(-\frac{\varepsilon^2 k}{8}\right) \end{aligned}$$

where we use that $(1 + \varepsilon)^2 \geq 1 + \varepsilon$ and $(1 - \varepsilon)^2 \leq 1 - \varepsilon$ for all $0 \leq \varepsilon \leq 1$. We note that $T(y)$ is a $\frac{1}{\sqrt{k}}$ -scaling of a standard Gaussian random vector and so the claim follows from Theorem 2.11. \square

Then applying the union bound and using Claim I for all difference vectors $x_i - x_j$ we obtain

$$\begin{aligned} \Pr\left[\exists i \neq j : \left| \frac{\|T(x_i) - T(x_j)\|_2}{\|x_i - x_j\|_2} - 1 \right| > \varepsilon\right] &\leq \sum_{1 \leq i < j \leq n} \Pr\left[\frac{\|T(x_i - x_j)\|_2}{\|x_i - x_j\|_2} > \varepsilon\right] \\ &\stackrel{\text{Claim I}}{\leq} n^2 \cdot 2 \exp(\varepsilon^2 k/8) \stackrel{k := \frac{8 \ln(4n^3)}{\varepsilon^2}}{=} \frac{1}{2n} \end{aligned}$$

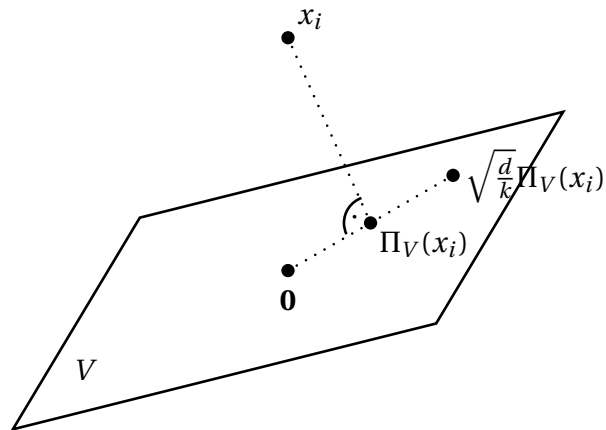
Note that we use linearity of T . \square

The reader may note that while the embedding guarantees to preserve that for the $\binom{n}{2}$ many difference vectors one has $\|T(x_i - x_j)\|_2 \approx \|x_i - x_j\|_2$, it is not possible to preserve the length of all vectors. As long as $k < d$, there will always be a non-zero vector u with $Gu = \mathbf{0}$ and so $\|T(u)\|_2 = 0$.

On the other hand one might wonder what other random embeddings $T : \mathbb{R}^d \rightarrow \mathbb{R}^k$ one could have used. For example one could pick a set $I \subseteq [d]$ of $|I| = k$ indices at random and set $T(x) := \sqrt{\frac{d}{k}} \cdot (x_i)_{i \in I}$. Note that each coordinate is selected with probability $\frac{k}{d}$ and so for any y one has $\mathbb{E}[\|T(y)\|_2^2] = \sum_{i=1}^d \Pr[i \in I] \cdot (\sqrt{\frac{d}{k}} y_i)^2 = \|y\|_2^2$. Hence this embedding also gets the expectation correct. But the length does not concentrate well. For example the vector $e_1 = (1, 0, \dots, 0)$ has

$$\|T(e_1)\|_2^2 = \begin{cases} \frac{d}{k} & \text{with probability } \frac{k}{d} \\ 0 & \text{otherwise} \end{cases}$$

A different construction that does work however is to choose a uniform random subspace $V \subseteq \mathbb{R}^d$ of dimension $k := \Theta\left(\frac{\ln(n)}{\varepsilon^2}\right)$, project the points and then scale by $\sqrt{\frac{d}{k}}$. That means T is defined by $T(y) := \sqrt{\frac{d}{k}} \cdot \Pi_V(y)$ where Π_V denotes the orthogonal projection into V .



While this geometric view is more intuitive, the analysis using a random Gaussian matrix is easier (because there all entries G_{ij} are independent).

Chapter 3

Algebraic algorithms

In this chapter we discuss algorithms that use *algebraic methods*. The most important object in this regard are *polynomials*.

Definition 3.1. A *monomial* in variables x_1, \dots, x_n is a product of the variables with non-negative integer exponents, times a constant coefficient. That means a monomial is of the form

$$c \cdot x_1^{a_1} \cdot \dots \cdot x_n^{a_n}$$

where $a_1, \dots, a_n \in \mathbb{Z}_{\geq 0}$ and $c \in \mathbb{R}$. The *degree*¹ of the monomial is the sum of all exponents, i.e. $a_1 + \dots + a_n$. A (*multivariate*) *polynomial* p is a finite sum of monomials, i.e. it is of the form

$$p(x_1, \dots, x_n) = \sum_a c_a \prod_{i=1}^n x_i^{a_i}$$

The *degree* $\deg(p)$ of a polynomial is the maximum degree of any of its monomials that have non-zero coefficient.

Note that a polynomial p is in particular a function² $p: \mathbb{R}^n \rightarrow \mathbb{R}$. If $n = 1$, then p is called a *univariate polynomial*.

Example 3.2. The polynomial $p(x_1, x_2) := 2x_1^2 + 3x_1x_2^2$ has degree $\deg(p) = 3$. It has two monomials, which are $2x_1^2$ and $3x_1x_2^2$.

¹In the literature this is sometimes called the *total degree* of a monomial. In contrast the *maximum degree* is defined to be $\max_{i=1, \dots, n} a_i$. In this chapter, we will not need both quantities, so we refrain from a distinction.

²In the most general setting one could let the coefficients c_a be from any field \mathbb{F} and then the polynomial would be a function $p: \mathbb{F}^n \rightarrow \mathbb{R}$. We do not need that generality so for the sake of simplicity we stick to real polynomials.

We mention a very important example of a polynomial that we revisit later:

Example 3.3. For a matrix $A \in \mathbb{R}^{n \times n}$, let us write $A = (A_{ij})_{i,j \in [n]}$. The *Leibniz formula* then says that the *determinant* of the matrix is

$$\det(A) = \sum_{\sigma: [n] \rightarrow [n]} \operatorname{sgn}(\sigma) \cdot \prod_{i=1}^n A_{i, \sigma(i)}$$

where the sum runs over all permutations from $[n]$ to $[n]$ and $\operatorname{sgn}(\sigma) \in \{-1, 1\}$ (one has $\operatorname{sgn}(\sigma) = +1$ if σ is obtained from the identity permutation by an even number of transpositions; but that won't be too important for us). We can conclude that $\det(A)$ is a polynomial of degree at most n where the variables are the n^2 entries³.

There is a very important polynomial that we give a distinguished name:

Definition 3.4. The *zero polynomial* 0 is the polynomial that has no monomials with non-zero coefficients. For two polynomials p and q we write $p \equiv q$ if they agree in all their monomials.

In particular we are interested in the following problem:

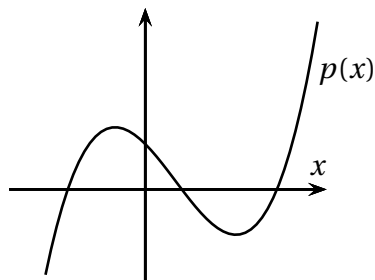
Polynomial identity testing. Given oracle access to a polynomial p in variables x_1, \dots, x_n . Decide whether $p \equiv 0$.

Here *oracle access* means that for a number of points $x \in \mathbb{R}^n$ we may query the value $p(x) \in \mathbb{R}$. Note that testing whether $p \equiv q$ is the same as testing whether $p - q \equiv 0$. To understand why oracle access may be a meaningful assumption, note that for any matrix A we can compute $\det(A)$ in times $O(n^3)$. On the other hand, considered as a polynomial, \det has $n!$ many monomials so we would not want to write all of them out.

That brings us to the following question: If p is the zero polynomial we clearly have $p(x_1, \dots, x_n) = 0$ for all $x_1, \dots, x_n \in \mathbb{R}$. But is the reverse true? In the univariate case, the answer follows from the following classic result:

Theorem 3.5. Let $p(x) = \sum_{i=0}^d c_i x^i$ be a univariate polynomial with $c_0, \dots, c_d \in \mathbb{R}$, $d = \deg(p) \geq 1$, then p has at most d many roots.

³For $n = 2$ one can verify that $A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$ has $\det(A) = A_{11}A_{22} - A_{21}A_{12}$.

degree-3 polynomial f over \mathbb{R} with 3 zeros

Hence for a univariate polynomial p with degree d one can do polynomial identity testing by querying $f(s_1), \dots, f(s_{d+1})$ for arbitrary (distinct) points $s_1, \dots, s_{d+1} \in \mathbb{R}$ and checking whether $p(s_i) = 0$ for all $i = 1, \dots, d + 1$. But for the multivariate case, the situation is more complicated. For example the polynomial $p(x_1, x_2) := 1 - x_1 x_2$ has infinitely many zeroes, yet it is not the zero polynomial. But a more careful argument will work. But first, to warm up we consider a simpler variant.

3.1 Matrix Identity testing

Suppose we have matrices $A, B, C \in \mathbb{R}^{n \times n}$ and we want to test if $AB = C$. Of course we could compute AB in time⁴ $O(n^3)$ and then check directly if $AB = C$. But maybe we do not need to actually compute the matrix product AB to know whether its equal to C . Suppose we select a large enough set $S \subseteq \mathbb{R}$ and we draw a vector $x = (x_1, \dots, x_n)$ by drawing each coordinate $x_i \sim S$ independently. Then we merely test whether

$$A(Bx) = Cx$$

This test can be done in time $O(n^2)$ since we only need three matrix-vector multiplications. But on the other hand the test can make a mistake; it could be that $ABx = Cx$ while $AB \neq C$. So it is important to prove that making such a mistake is unlikely.

Theorem 3.6. *Let $A, B, C \in \mathbb{R}^{n \times n}$ be any matrices with $AB \neq C$ and let $S \subseteq \mathbb{R}$. Then*

$$\Pr_{x_i \sim S} [ABx = Cx] \leq \frac{1}{|S|}$$

Proof. We set $D := AB \in \mathbb{R}^{n \times n}$. If $D \neq C$, then there has to be at least one row i so that $D_i \neq C_i$. It suffices to show that for this fixed row one has $\Pr[\langle D_i, x \rangle =$

⁴It can be done a faster using Strassen's algorithm in time $O(n^{2.8074})$. The current record is $O(n^{2.371552})$.

$\langle C_i, x \rangle \leq \frac{1}{|S|}$. There has to be one $j \in [n]$ so that $D_{ij} \neq C_{ij}$. Let us fix any outcome of $x_1, \dots, x_{j-1}, x_{j+1}, \dots, x_n$. Then using only the randomness over x_j we have

$$\begin{aligned} \Pr_{x_j \sim S} [\langle D_i, x \rangle = \langle C_i, x \rangle] &= \Pr_{x_j \sim S} \left[(D_{ij} - C_{ij})x_j = \sum_{\ell \neq j} (C_{i\ell} - D_{i\ell})x_\ell \right] \\ &= \Pr_{x_j \sim S} \left[x_j = \underbrace{\frac{1}{D_{ij} - C_{ij}} \sum_{\ell \neq j} (C_{i\ell} - D_{i\ell})x_\ell}_{=: \alpha} \right] \leq \frac{1}{|S|} \end{aligned}$$

In the last step we use the following crucial argument: there is only one value for x_j that makes the equation $x_j = \alpha$ true. Hence if we draw $x_j \sim S$, the chance of hitting α is at most $\frac{1}{|S|}$. \square

The argument used above is a simple cases of the *principle of deferred decision*. In a randomized algorithm it can be useful to not draw all random bits at the beginning but rather imagine to only generate them when their value becomes relevant. Similarly in the probabilistic proof above we imagined to draw the random variable x_j last which makes the proof much easier.

Finally we would like to point out that the proof of Theorem 3.6 implicitly gives an upper bound on the number of roots of $p(x) := \langle D_i - C_i, x \rangle$ which is a multivariate polynomial of degree 1.

3.2 The Schwarz Zippel Lemma

We now go back to the question of how to limit the number of zeroes of a non-zero multivariate polynomial. The trick is to restrict to a *combinatorial rectangle* which is a set of the form $S \times S \times \dots \times S$ for some set $S \subseteq \mathbb{R}$.

Lemma 3.7 (Schwarz-Zippel 1979). *Let $p(x_1, \dots, x_n)$ be a polynomial that is not the zero polynomial and let $d := \deg(p)$ be its degree. Let $S \subseteq \mathbb{R}$ be a finite set. Then*

$$\Pr_{a_1, \dots, a_n \sim S} [p(a_1, \dots, a_n) = 0] \leq \frac{d}{|S|}$$

Note that here a_1, \dots, a_n are chosen independently at random from S .

Proof. We prove the claim by induction over $n \geq 1$. For $n = 1$ we know by Theorem 3.5 that p has at most d roots and the claim follows.

Now consider a polynomial $p(x_1, \dots, x_{n+1})$ where $n \geq 1$ and let $d := \deg(p)$ be its degree. We pull out the last variable x_{n+1} from each monomial of p ; then we

can write

$$p(x_1, \dots, x_{n+1}) = \sum_{i=0}^{\ell} \underbrace{\left(\sum_{a \in \mathbb{Z}_{\geq 0}^{n+1}: a_{n+1}=i} c_a \prod_{j=1}^n x_j^{a_j} \right)}_{=: p_i(x_1, \dots, x_n)} \cdot x_{n+1}^i = \sum_{i=0}^{\ell} p_i(x_1, \dots, x_n) \cdot x_{n+1}^i$$

where each p_i is a polynomial with n variables and degree $\deg(p_i) \leq d - i$. The number ℓ is the largest power so that x_{n+1}^{ℓ} appears in any non-zero monomial.

Claim I. Fix any $a_1, \dots, a_n \in \mathbb{R}$ so that $p_{\ell}(a_1, \dots, a_n) \neq 0$. Then

$$\Pr_{a_{n+1} \sim S} [p(a_1, \dots, a_{n+1}) = 0] \leq \frac{\ell}{|S|}.$$

Proof of Claim I. Fix a_1, \dots, a_n with $p_{\ell}(a_1, \dots, a_n) \neq 0$. Consider the univariate polynomial q defined by $q(x_{n+1}) := p(a_1, \dots, a_n, x_{n+1})$. Then $\deg(q) = \ell$ and by assumption q is not the zero-polynomial. Then q has at most ℓ roots by Theorem 3.5. \square

We continue with the main proof. By Claim I we know that a zero is unlikely if the highest term p_{ℓ} did not vanish. But that term p_{ℓ} is a polynomial with n variables so by induction it is unlikely to vanish. Putting everything together we get:

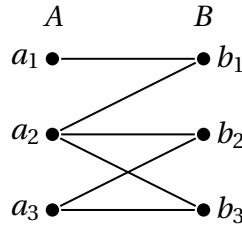
$$\begin{aligned} & \Pr_{a_1, \dots, a_{n+1} \sim S} [p(a_1, \dots, a_{n+1}) = 0] \\ \leq & \underbrace{\Pr_{a_1, \dots, a_n \sim S} [p_{\ell}(a_1, \dots, a_n) = 0]}_{\leq \frac{\deg(p_{\ell})}{|S|} \text{ by induction}} + \underbrace{\Pr_{a_1, \dots, a_{n+1} \sim S} [p(a_1, \dots, a_{n+1}) = 0 \mid p_{\ell}(a_1, \dots, a_n) \neq 0]}_{\leq \frac{\ell}{|S|} \text{ by Claim I}} \\ \leq & \frac{d - \ell}{|S|} + \frac{\ell}{|S|} = \frac{d}{|S|} \end{aligned}$$

\square

This answers the question that we discussed earlier: given a polynomial p in variables x_1, \dots, x_n with (total) degree d , we can select any set $S \subseteq \mathbb{R}$ of size $|S| \geq 2d$ and query a random point x with $x_1, \dots, x_n \sim S$. Then if p is not the zero polynomial, we have $\Pr[p(x) \neq 0] \geq \frac{1}{2}$. This gives a randomized algorithm for polynomial identity testing in the oracle model.

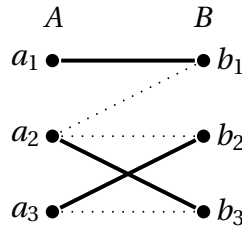
3.3 Bipartite matchings

We will now describe a rather magical application of polynomial identity testing. Consider a *bipartite graph* $G = (A \dot{\cup} B, E)$ with $|A| = |B| = n$. Recall that “bipartite” means that all edges $e \in E$ run between A and B . It will be convenient to denote the vertices on both sides as $A = \{a_1, \dots, a_n\}$ and $B = \{b_1, \dots, b_n\}$.



example of a bipartite graph

We are interested in the question whether G contains a *perfect matching* $F \subseteq E$, which is a set of edges so that each node $v \in A \cup B$ is incident to exactly one edge.



perfect matching in bold

Using the concept of augmenting paths one can construct a polynomial time algorithm that finds a perfect matching in G if there is any [Edm65]. With some more work the running time can even be brought down to $O(|E|\sqrt{n})$ [MV80, Vaz20] (both algorithms work even in general graphs, not just bipartite ones). Our goal here is to describe a randomized algorithm instead.

The *adjacency matrix* $M \in \{0, 1\}^{n \times n}$ of G is defined by

$$M(i, j) := \begin{cases} 1 & \text{if } \{a_i, b_j\} \in E \\ 0 & \text{otherwise} \end{cases}$$

Next, we take the adjacency matrix M and replace every 1 by a *variable* x_{ij} and call the new matrix M_x . That means the entries of the matrix M_x are

$$M_x(i, j) := \begin{cases} x_{ij} & \text{if } \{a_i, b_j\} \in E \\ 0 & \text{otherwise} \end{cases}$$

Then from Example 3.3 we know that $\det(M_x)$ is a multivariate polynomial with $|E|$ many variables. Let us revisit the example above. In that case we have

$$M = \begin{matrix} & \begin{matrix} b_1 & b_2 & b_3 \end{matrix} \\ \begin{matrix} a_1 \\ a_2 \\ a_3 \end{matrix} & \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \end{matrix} \quad M_x = \begin{matrix} & \begin{matrix} b_1 & b_2 & b_3 \end{matrix} \\ \begin{matrix} a_1 \\ a_2 \\ a_3 \end{matrix} & \begin{bmatrix} x_{1,1} & 0 & 0 \\ x_{2,1} & x_{2,2} & x_{2,3} \\ 0 & x_{3,2} & x_{3,3} \end{bmatrix} \end{matrix}$$

and the determinant is

$$\det(M_x) = x_{1,1} \cdot x_{2,2} \cdot x_{3,3} - x_{1,1} \cdot x_{2,3} \cdot x_{3,2}$$

We observe that (at least in this example) the monomials of $\det(M_x)$ happen to correspond to perfect matchings in G . We will prove that this is not a coincidence:

Theorem 3.8. *Let $G = (A \cup B, E)$ be a bipartite graph with $|A| = |B|$. Then G has a perfect matching if and only if $\det(M_x)$ is not the zero polynomial.*

Proof. First we make an observation: a permutation $\sigma : [n] \rightarrow [n]$ corresponds to a perfect matching whose edges we can write as $F(\sigma) := \{\{a_i, b_{\sigma(i)}\} : i \in [n]\}$ and vice versa. We recall the Leibniz formula for the determinant:

$$\det(M_x) = \sum_{\sigma: [n] \rightarrow [n]} \operatorname{sgn}(\sigma) \prod_{i=1}^n M_x(i, \sigma(i)) \stackrel{(*)}{=} \sum_{\substack{\sigma: [n] \rightarrow [n]: \\ F(\sigma) \subseteq E}} \operatorname{sgn}(\sigma) \cdot \prod_{i=1}^n x_{i, \sigma(i)}$$

Here the first sum runs over all permutations $\sigma : [n] \rightarrow [n]$ and the second sum runs only over those permutations whose edge set is in E . The argument for $(*)$ is the following: for any permutation σ where $F(\sigma) \not\subseteq E$, there is an index i with $\{a_j, b_{\sigma(j)}\} \notin E$ and so the product $\prod_{i=1}^n M_x(i, \sigma(i))$ contains a 0. So the only terms left are those corresponding to perfect matchings contained in the graph and for those, one has $\prod_{i=1}^n M_x(i, \sigma(i)) = \prod_{i=1}^n x_{i, \sigma(i)}$. Now consider both directions of the statement.

(\Rightarrow) : Suppose there is a perfect matching corresponding to a permutation σ . Then $\operatorname{sgn}(\sigma) \prod_{i=1}^n x_{i, \sigma(i)}$ is a non-zero monomial appearing in $\det(M_x)$; note that no other permutation has the same product $\prod_{i=1}^n x_{i, \sigma(i)}$, hence there are no cancellations.

(\Leftarrow) : If $\det(M_x)$ has a non-zero monomial, then it has to be of the form $\operatorname{sgn}(\sigma) \cdot \prod_{i=1}^n x_{i, \sigma(i)}$ with $F(\sigma) \subseteq E$. That means $F(\sigma)$ is a perfect matching in G . \square

Now consider the following algorithm:

PERFECT MATCHING TEST IN BIPARTITE GRAPHS

Input: Bipartite graph G .

Decide: Does G contain a perfect matching.

- (1) Independently draw $y_{ij} \sim \{1, \dots, n^2\}$ for all $\{a_i, b_j\} \in E$
- (2) IF^a $\det(M_x)(y) \neq 0$ then return “YES” else return “NO”.

^aWe mean that we evaluate the polynomial $p(x) := \det(M_x)$ at point y

Then combining the Schwarz-Zippel Lemma (Lemma 3.7) with Theorem 3.8 we can see that the test always rejects if G does not contain a perfect matching and if G does contain a perfect matching, then it accepts with probability at least $1 - \frac{1}{n}$ (since $\det(M_x)$ has degree n and $|S| = n^2$). Note that the algorithm runs in time $O(n^3)$, so it does not actually beat the best deterministic algorithm. But the current algorithm is *parallelizable* which is often desirable. Let us say that an algorithm is a *fast parallel algorithm* if on input length n it runs in time $\text{poly}(\log(n))$ when it is allowed to use $\text{poly}(n)$ many processors. We will not go much into details what that means. But we want to quote the following essential result:

Theorem 3.9 ([BvH82]). *There is a (deterministic) fast parallel algorithm to compute $\det(A)$ for any $A \in \mathbb{Q}^{n \times n}$.*

Since testing whether the determinant of a matrix is 0 is the main work to be done in our algorithm we may conclude:

Theorem 3.10 ([Mul86]). *Given a bipartite graph G , there there is a fast (randomized) algorithm to test whether G contains a perfect matching.*

One aspect of the algorithm that is less satisfying is that it only answers whether or not there *exists* a perfect matching in a graph — it does not actually produce that matching. However one could use the test as a black box and make it a search algorithm as follows: Let $E = \{e_1, \dots, e_m\}$. Test if E contains a perfect matching. If so, delete e_1 , test again. If now the answer is negative, then add e_1 back. Proceed with e_2 and so on. But this gives a highly sequential algorithm and it does not seem easy to parallelize it. For example we could test in parallel for all i whether $E \setminus \{e_i\}$ contains a perfect matching. But it can happen that each single edge is expendable and the answer is always “YES” which would not be very helpful.

But there is a more elegant approach. For the moment let us assume that each edge $e \in E$ has a weight $2^{w(e)}$ assigned where $w(e) \in \mathbb{N}$ and furthermore let us assume that there is a unique perfect matching $F^* \subseteq E$ that minimizes the weight $\sum_{e \in F^*} 2^{w(e)}$ (we will later justify where those weights are coming from). Then the determinant at the point $y \in \mathbb{R}^E$ with $y_e := 2^{w(e)}$ has a value of

$$\det(M_x)(y) = \sum_{\substack{\sigma: [n] \rightarrow [n]: \\ F(\sigma) \subseteq E}} \text{sign}(\sigma) \underbrace{\prod_{e \in F(\sigma)} 2^{w(e)}}_{=2^{w(F(\sigma))}} = 2^{w(F^*)} \cdot (\pm 1 + \text{an even number})$$

where we abbreviate $w(F^*) = \sum_{e \in F^*} w(e)$. That means $\det(M_x)(y)$ is an integer multiple of $2^{w(F^*)}$ (we note that it was crucial that there was only one perfect

matching with this value). Then we can use the fast parallel algorithm from Theorem 3.9 to compute $\det(M_x)(y)$ and then infer the value of $w(F^*)$. That does not immediately tell us the edge set F^* . But we can now run the algorithm in parallel for each $e \in E$ and test whether the value of the minimum matching has changed for the edge set $E \setminus \{e\}$ ⁵. After this we know precisely all the edges in F^* .

That leaves us with the question where the weights $w(e)$ come from in the first place. We use a remarkable result by Mulmuley, Vazirani, and Vazirani which says that in basically any optimization problem, if we choose random weights from a polynomially large range, then the minimum weight solution will likely be unique.

Theorem 3.11 (Isolation lemma [MVV87]). *Let \mathcal{F} of any family of subsets of $\{1, \dots, n\}$ and let $W \in \mathbb{N}$. Draw independently random weights $w_i \sim \{1, \dots, W\}$ for each $i \in [n]$. Then the problem*

$$\min\{w(S) : S \in \mathcal{F}\}$$

has a unique optimum solution with probability at least $1 - \frac{n}{W}$.

Note that \mathcal{F} can have up to 2^n many sets and the solutions have weight $w(S) \in \{0, \dots, nW\}$. Hence by the *pigeonhole principle*, there always will be *some* value that is attained by at least $\frac{|\mathcal{F}|}{nW+1}$ many solutions — just that this won't be the minimum value.

Now back to our perfect matching problem. If we draw $w(e) \sim \{1, \dots, 2|E|\}$ for each edge $e \in E$, then with probability at least $1/2$, the minimum perfect matching will be unique. That concludes the argument and we can derive:

Theorem 3.12. *Given a bipartite graph G , there is a fast parallel algorithm to find a perfect matching if there is any.*

3.4 Perfect matchings in general graphs

Now, let $G = (V, E)$ be an arbitrary (i.e. not necessarily bipartite) undirected graph. We write $V = \{a_1, \dots, a_n\}$. Again we are interested in determining whether G contains a perfect matching. It turns out that the method behind Theorem 3.8 can be generalized, but it takes a lot more care. First, we change the definition of

⁵There is the complication that if $e \in F^*$, then there may be many matchings of the now-minimal value. Still we can find the minimum $W \in \mathbb{Z}_{\geq 0}$ so that $\det(M_x)(y) = 2^W \cdot \text{odd number}$ and test if $w(F^*) = W$.

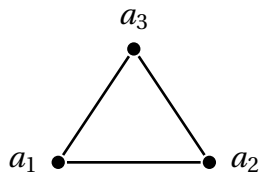
the matrix to \tilde{M}_x with

$$\tilde{M}_x(i, j) := \begin{cases} x_{ij} & \text{if } \{a_i, a_j\} \in E \text{ and } i < j \\ -x_{ij} & \text{if } \{a_i, a_j\} \in E \text{ and } i > j \\ 0 & \text{otherwise} \end{cases}$$

Note that the matrix \tilde{M}_x is *skew-symmetric* (i.e. $\tilde{M}_x^T = -\tilde{M}_x$). Then the following holds:

Theorem 3.13 (Tutte [Tut47]). *Graph G contains a perfect matching if and only if $\det(\tilde{M}_x)$ is not the zero polynomial.*

We omit the proof due to time constraints⁶. But we want to point out the difficulty that arises. Consider the following example of a triangle graph:



$$\tilde{M}_x = \begin{array}{ccc|c} & a_1 & a_2 & a_3 & \\ \hline & \mathbf{0} & x_{\{1,2\}} & x_{\{1,3\}} & a_1 \\ -x_{\{1,2\}} & & \mathbf{0} & x_{\{2,3\}} & a_2 \\ -x_{\{1,3\}} & -x_{\{2,3\}} & & \mathbf{0} & a_3 \end{array}$$

Of course the triangle graph does not contain a perfect matching and indeed $\det(\tilde{M}_x) \equiv 0$. But the product $x_{\{1,2\}} \cdot x_{\{1,3\}} \cdot x_{\{2,3\}}$ appears several times in the Leibniz formula and only the arising cancellations lead the right result.

⁶The proof can be found for example in the notes by Schulman, Theorem 38 in <http://users.cms.caltech.edu/~schulman/Courses/18cs150/lec11.pdf>

Chapter 4

Linear algebra

In this chapter we will review some linear algebra results and we will learn some algorithmic applications. For more in-depth treatment of the linear algebra part we recommend the textbook by Horn and Johnson [HJ90].

4.1 Eigenvalues

For a matrix $A \in \mathbb{R}^{n \times n}$ we say that (λ, x) with $\lambda \in \mathbb{R}$ and a vector $x \in \mathbb{R}^n$ is a (*real*) *Eigenvalue-Eigenvector pair* if

$$Ax = \lambda x$$

In the following we write I_n as the $n \times n$ identity matrix. We require the following definition:

Definition 4.1. The univariate polynomial $\det(xI_n - A)$ in variable $x \in \mathbb{R}$ is called the *characteristic polynomial* of $A \in \mathbb{R}^{n \times n}$.

Recall that $\det(xI_n - A)$ is indeed a polynomial by the Leibniz formula. The connection to Eigenvalues is the following:

Theorem 4.2. Let $A \in \mathbb{R}^{n \times n}$. The Eigenvalues of A are exactly the roots of the characteristic polynomial $\det(xI_n - A)$.

Proof. Fix $\lambda \in \mathbb{R}$. Then

$$\begin{aligned} \det(\lambda I_n - A) = 0 &\Leftrightarrow \text{rank}(\lambda I_n - A) < n \\ &\Leftrightarrow \exists v \in \mathbb{R}^n \setminus \{\mathbf{0}\} : (\lambda I_n - A)v = \mathbf{0} \\ &\Leftrightarrow \exists v \in \mathbb{R}^n \setminus \{\mathbf{0}\} : \lambda v = Av \end{aligned}$$

That means λ is a root of the characteristic polynomial if and only if it is an Eigenvalue together with some vector v . \square

We note that without making any assumptions on A , the matrix may not even have any (real) Eigenvalues. We could extend the definition to *complex Eigenvalue-Eigenvector pairs* (λ, v) with $\lambda \in \mathbb{C}$ and $v \in \mathbb{C}^n$. In that case the Fundamental Theorem of Calculus (Gauss 1799) yields that there are exactly n roots of the polynomial $\det(xI_n - A)$ (if counted with multiplicity; the roots may be complex) which provide exactly n Eigenvalue-Eigenvector pairs¹.

4.1.1 The Spectral Theorem

An matrix $A \in \mathbb{R}^{n \times n}$ is called *symmetric* if $A_{ij} = A_{ji}$ for all i, j . This gives a well-behaved class of matrices that are algorithmically very relevant (for example adjacency matrices of undirected graphs are symmetric) and have all Eigenvalues real.

Theorem 4.3 (Spectral Theorem). *For any symmetric matrix $A \in \mathbb{R}^{n \times n}$ there are Eigenvalues $\lambda_1, \dots, \lambda_n \in \mathbb{R}$ with corresponding orthonormal Eigenvectors $v_1, \dots, v_n \in \mathbb{R}^n$. In particular*

$$M = \sum_{i=1}^n \lambda_i v_i v_i^T$$

This gives a very useful decomposition of any symmetric matrix that is also called the *Eigen decomposition*. For example for any vector $x \in \mathbb{R}^n$ we have

$$Mx = \sum_{i=1}^n \lambda_i v_i \langle v_i, x \rangle$$

which means that a matrix multiplication with M means we stretch the part of x that goes in direction v_i by a factor of λ_i .

4.1.2 Positive semidefinite matrices

An important class of matrices are those with non-negative Eigenvalues.

¹For example $A = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$ has the characteristic polynomial $\det(xI_2 - A) = x^2 + 1$ which has the complex roots i and $-i$. The corresponding (complex) Eigenvectors are $\begin{pmatrix} i \\ 1 \end{pmatrix}, \begin{pmatrix} -i \\ 1 \end{pmatrix}$.

Definition 4.4. Let $A \in \mathbb{R}^{n \times n}$ be a symmetric matrix with Eigenvalues $\lambda_1, \dots, \lambda_n \in \mathbb{R}$. We say that A is *positive semidefinite* if $\lambda_i \geq 0$ for all $i = 1, \dots, n$. We write $A \geq 0$ iff A is positive semidefinite. More generally for two symmetric matrices $A, B \in \mathbb{R}^{n \times n}$ we write $A \geq B : \Leftrightarrow A - B \geq 0$.

Note that if we write $A \geq 0$ we always implicitly mean that A is symmetric, too. Note that \geq is a *partial order* called the *Löwner ordering*. Using the spectral Theorem it is relatively easy to derive the following:

Lemma 4.5. For a symmetric matrix $A \in \mathbb{R}^{n \times n}$, the following is equivalent

- a) $A \geq 0$
- b) $x^T A x \geq 0 \forall x \in \mathbb{R}^n$.
- c) There exists a matrix U so that $A = U U^T$.

4.1.3 A geometric interpretation

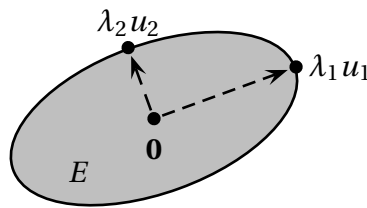
Let $A \in \mathbb{R}^{n \times n}$ be a symmetric, positive definite matrix. Consider the set

$$E := \{x \in \mathbb{R}^n \mid x^T A^{-2} x \leq 1\}$$

which is also called an *ellipsoid*. Using the Eigen decomposition $A = \sum_{i=1}^n \lambda_i u_i u_i^T$ we can rewrite

$$E = \left\{ x \in \mathbb{R}^n \mid \sum_{i=1}^n \frac{1}{\lambda_i^2} \langle u_i, x \rangle^2 \leq 1 \right\}$$

In particular, the points $x = \lambda_i u_i$ satisfy the inequality with equality which means that these points lie on the boundary of E . In fact, the directions u_1, \dots, u_n denote the *axis* of the ellipsoid and λ_i is the *length* of the i th axis.



Example for $n = 2$

If we substitute $y := A^{-1} x \Leftrightarrow x = A y$ then we can write alternatively

$$E = \{A y \mid y \in \mathbb{R}^n \text{ with } \|y\|_2 \leq 1\}$$

meaning that E is the image of the Euclidean ball under the map $x \mapsto A x$. One can argue that

$$\text{Vol}_n(E) = \left(\prod_{i=1}^n \lambda_i \right) \cdot \text{Vol}_n(B_2^n) = \det(A) \cdot \text{Vol}(B_2^n)$$

where we will prove the second equality later.

4.1.4 Applying functions to matrices

The Eigen decomposition allows us to apply arbitrary univariate functions to matrices:

Definition 4.6. Let $A \in \mathbb{R}^{n \times n}$ be a symmetric matrix with Eigen decomposition $A = \sum_{i=1}^n \lambda_i v_i v_i^T$. For any function $f : \mathbb{R} \rightarrow \mathbb{R}$ we define

$$f(A) := \sum_{i=1}^n f(\lambda_i) v_i v_i^T$$

It turns out that this definition aligns with many matrix functions that we already know. Consider a symmetric matrix A with Eigen decomposition $A = \sum_{i=1}^n \lambda_i v_i v_i^T$. Now, the following holds:

- For $k \in \mathbb{Z}_{\geq 0}$, let $A^k = AA \dots A$ be the k -fold matrix product (i.e. $A^0 = I_n$, $A^1 = A$, $A^2 = AA$ and so on). Then $A^k = \sum_{i=1}^n \lambda_i^k v_i v_i^T$.
- The *inverse* of A is $A^{-1} = \sum_{i=1}^n \frac{1}{\lambda_i} v_i v_i^T$.

We can also use this notion to define new matrix functions that are very useful:

- The *matrix exponential* is

$$\exp(A) := \sum_{i=1}^n \exp(\lambda_i) v_i v_i^T$$

With the series representation for the 1-dimensional exponential function one can then verify that

$$\exp(A) = \sum_{k=0}^{\infty} \frac{A^k}{k!}$$

- If $A \geq 0$, then we define the *square root* as

$$\sqrt{A} = A^{1/2} := \sum_{i=1}^n \sqrt{\lambda_i} v_i v_i^T$$

Note that $A^{1/2} A^{1/2} = A$ as one would expect.

4.1.5 Trace, determinant and rank

A useful quantity for a square matrix is the sum of its diagonal entries:

Definition 4.7. For $A \in \mathbb{R}^{n \times n}$ we define the *trace* as $\text{Tr}[A] := \sum_{i=1}^n A_{ii}$.

Typically matrix multiplication does not commute. But the trace function has the following very useful property:

Lemma 4.8 (Cyclicity of the trace). *For any matrices A_1, \dots, A_k of any format so that the matrix product $A_1 A_2 \dots A_k$ is well-defined one has*

$$\text{Tr}[A_1 A_2 \dots A_k] = \text{Tr}[A_2 \dots A_k A_1]$$

One can easily prove this from the definition of trace and the observation that it suffices to prove this property for $k = 2$. The following is easy to see but we state it explicitly for later reference:

Lemma 4.9 (Linearity of the trace). *For any matrices $A, B \in \mathbb{R}^{n \times n}$ and $\lambda \in \mathbb{R}$ one has $\text{Tr}[A + B] = \text{Tr}[A] + \text{Tr}[B]$ and $\text{Tr}[\lambda A] = \lambda \text{Tr}[A]$.*

We can prove the following:

Lemma 4.10. *For any symmetric $A \in \mathbb{R}^{n \times n}$, $\text{Tr}[A]$ equals the sum of the Eigenvalues.*

Let $A = \sum_{i=1}^n \lambda_i v_i v_i^T$ be the Eigen decomposition of A which exists by Theorem 4.3. We note that in general the diagonal entries of A are *not* Eigenvalues — just that somehow their sum equals the sum of Eigenvalues. To strengthen our linear algebra skills, we provide two proofs for this lemma:

Proof 1. Let e_i be the i th standard basis vector in \mathbb{R}^n . Then

$$\text{Tr}[A] = \sum_{i=1}^n \overbrace{e_i^T A e_i}^{=A_{ii}} = \sum_{i=1}^n e_i^T \left(\sum_{j=1}^n \lambda_j v_j v_j^T \right) e_i = \sum_{j=1}^n \lambda_j \underbrace{\sum_{i=1}^n \langle e_i, v_j \rangle^2}_{=\|v_j\|_2^2=1} = \sum_{j=1}^n \lambda_j$$

□

In fact, the same proof also gives that for *any* orthonormal basis u_1, \dots, u_n , one has that $\sum_{i=1}^n u_i^T A u_i = \sum_{j=1}^n \lambda_j$.

Proof 2. Using the linearity and cyclicity of the trace we get

$$\operatorname{Tr}[A] = \operatorname{Tr}\left[\sum_{i=1}^n \lambda_i v_i v_i^T\right] \stackrel{\text{linearity}}{=} \sum_{i=1}^n \lambda_i \operatorname{Tr}[v_i v_i^T] \stackrel{\text{cyclicity}}{=} \sum_{i=1}^n \lambda_i \operatorname{Tr}\left[\underbrace{v_i^T v_i}_{=\|v\|_2^2=1}\right] = \sum_{i=1}^n \lambda_i$$

We note that for a vector $v \in \mathbb{R}^n$, $v v^T$ is $n \times n$ rank-1 matrix while $v^T v \in \mathbb{R}$ is just a number. Yet their trace is the same using the cyclicity property. \square

We can also prove the lemma without the symmetry assumption as long as we consider complex Eigen values².

Lemma 4.11. *Let $A \in \mathbb{R}^{n \times n}$ be a matrix with (possibly complex) Eigenvalues $\lambda_1, \dots, \lambda_n \in \mathbb{C}$. Then $\operatorname{Tr}[A] = \sum_{i=1}^n \lambda_i$.*

The difficulty lies in the fact that the Spectral Theorem does not apply anymore and we cannot use the Eigen decomposition in the proof.

Proof. As seen earlier the Eigenvalues $\lambda_1, \dots, \lambda_n \in \mathbb{C}$ are the roots of the characteristic polynomial. Expanding that polynomial we can see that

$$\det(xI_n - A) = \prod_{i=1}^n (x - \lambda_i) = x^n + x^{n-1} \cdot (-(\lambda_1 + \dots + \lambda_n)) + (\text{terms of order } 0, \dots, n-2)$$

So it suffices to justify that the coefficient of x^{n-1} in the characteristic polynomial is $-\operatorname{Tr}[A]$. The other polynomial representation that we know for the determinant is the Leibniz formula. Hence

$$\det(xI_n - A) = \sum_{\sigma: [n] \rightarrow [n]} \operatorname{sign}(\sigma) \prod_{i=1}^n (xI_n - A)_{i, \sigma(i)}$$

We observe that those permutations σ with less than $n-1$ many fix points cannot contribute any term of the form x^{n-1} . Moreover there are no permutations with exactly $n-1$ fix points and there is only one permutation with n fix points — the identity permutation. Then for the identity permutation (which has sign 1) we get a contribution of

$$\prod_{i=1}^n (xI_n - A)_{i,i} = \prod_{i=1}^n (x - A_{ii}) = x^n + x^{n-1} \underbrace{(-A_{11} - A_{22} - \dots - A_{nn})}_{-\operatorname{Tr}[A]} + (\text{terms of ord. } \leq n-2)$$

²The statement may seem odd as it suggests that the real number $\operatorname{Tr}[A]$ equals a sum of complex numbers. But $\lambda_1, \dots, \lambda_n \in \mathbb{C}$ are the roots of a polynomial with real coefficients and so the roots come in pairs of complex conjugates, i.e. if we have a root of the form $a + bi$ ($a, b \in \mathbb{R}$) then there must be another root of the form $a - bi$ and their sum is $(a + bi) + (a - bi) = 2a$, meaning that the complex parts cancel out in the sum.

to Leibniz formula. Hence the coefficient of x^{n-1} in the characteristic polynomial is indeed $-\text{Tr}[A]$ and the claim follows. \square

The following is useful as well (we claimed it already in Sec 4.1.3):

Lemma 4.12. *For any matrix $A \in \mathbb{R}^{n \times n}$ with (possibly complex) Eigenvalues $\lambda_1, \dots, \lambda_n \in \mathbb{C}$, one has $\det(A) = \prod_{i=1}^n \lambda_i$.*

Proof. Recall that the characteristic polynomial is $\det(xI_n - A) = \prod_{i=1}^n (x - \lambda_i)$. Evaluating this at $x := 0$ gives $\det(-A) = \prod_{i=1}^n (-\lambda_i)$. Dividing both sides by $(-1)^n$ gives the claim. \square

4.1.6 Raleigh Quotient

In the following, let $A \in \mathbb{R}^{n \times n}$ be a symmetric matrix, which in particular means that A has n real Eigenvalues. It is common to write those Eigenvalues in ordered form as

$$\lambda_1(A) \geq \lambda_2(A) \geq \dots \geq \lambda_n(A)$$

(it is also common to write $\lambda_{\max}(A) := \lambda_1(A)$ and $\lambda_{\min}(A) := \lambda_n(A)$). There is a useful characterization of all such Eigenvalues as an optimization problem.

Theorem 4.13 (Raleigh Quotient). *For any symmetric matrix $A \in \mathbb{R}^{n \times n}$ one has*

$$\lambda_1(A) = \max_{\|x\|_2=1} x^T A x = \max_{x \in \mathbb{R}^n \setminus \{0\}} \frac{x^T A x}{x^T x}$$

Proof. Since A is symmetric, it admits an Eigen decomposition as $A = \sum_{i=1}^n \lambda_i u_i u_i^T$ where $\lambda_i := \lambda_i(A)$ is the i th Eigenvalue and u_1, \dots, u_n are an orthonormal basis of \mathbb{R}^n . Then for any x with $\|x\|_2 = 1$ one has

$$x^T A x = x^T \left(\sum_{i=1}^n \lambda_i u_i u_i^T \right) x = \sum_{i=1}^n \lambda_i \langle u_i, x \rangle^2$$

Since the u_i 's form an orthonormal basis we have $\sum_{i=1}^n \langle u_i, x \rangle^2 = \|x\|_2^2 = 1$ and so the expression $\sum_{i=1}^n \lambda_i \langle u_i, x \rangle^2$ is maximized when $|\langle u_1, x \rangle| = 1$ and $\langle u_i, x \rangle = 0$ for $i \neq 1$. Then the value is λ_1 . \square

More generally, the i th Eigenvalue is the maximum of $x^T A x$ subject to $\|x\|_2 = 1$ and x being orthogonal to u_1, \dots, u_{i-1} .

Theorem 4.14 (Courant Minimax Principle). *Let $A \in \mathbb{R}^{n \times n}$ be a symmetric matrix with Eigen decomposition $A = \sum_{i=1}^n \lambda_i u_i u_i^T$ where $\lambda_1 \geq \dots \geq \lambda_n$. Then for any $i \in [n]$ one has*

$$\lambda_i = \max_{\substack{\|x\|_2=1 \\ x \perp u_1, \dots, u_{i-1}}} x^T A x = \min_{v_1, \dots, v_{i-1} \in \mathbb{R}^n} \max_{\|x\|_2=1} x^T A x$$

We skip the proof which is basically an extension of the proof of Theorem 4.13. One can also directly characterize the *minimum* Eigenvalue:

Theorem 4.15 (Raleigh Quotient II). *For any symmetric matrix $A \in \mathbb{R}^{n \times n}$ one has*

$$\lambda_n(A) = \min_{\|x\|_2=1} x^T A x$$

From Theorem 4.15 we could also derive that a symmetric matrix $A \in \mathbb{R}^{n \times n}$ is positive-semidefinite if and only if $x^T A x \geq 0$ for all $x \in \mathbb{R}^n$ (though we already know this from Lemma 4.5). Again Theorem 4.15 can be extended to express any eigenvalue:

Theorem 4.16 (Courant Minimax Principle II). *Let $A \in \mathbb{R}^{n \times n}$ be a symmetric matrix with Eigen decomposition $A = \sum_{i=1}^n \lambda_i u_i u_i^T$ where $\lambda_1 \geq \dots \geq \lambda_n$. Then for any $i \in [n]$ one has*

$$\lambda_{n+1-i} = \min_{\substack{\|x\|_2=1 \\ x \perp u_{n-i+2}, \dots, u_n}} x^T A x = \max_{v_1, \dots, v_{i-1} \in \mathbb{R}^n} \min_{\substack{\|x\|_2=1 \\ x \perp v_1, \dots, v_{i-1}}} x^T A x$$

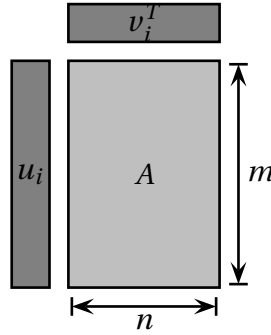
4.2 The Singular Value Decomposition

As we have seen, the Eigen decomposition is very useful in proofs as well as in designing algorithms. Unfortunately it only exists for symmetric matrices. It would be very useful to have a similar type of decomposition for an arbitrary matrix $A \in \mathbb{R}^{m \times n}$ (which may not even be a square matrix). And in fact, such a decomposition exists!

Theorem 4.17 (Singular Value Decomposition (SVD)). *Any matrix $A \in \mathbb{R}^{m \times n}$ can be written in the form*

$$A = \sum_{i=1}^r \sigma_i u_i v_i^T$$

where (i) $r = \text{rank}(A)$, (ii) $\sigma_1 \geq \dots \geq \sigma_r > 0$ are the so-called singular values, (iii) $u_1, \dots, u_r \in \mathbb{R}^m$ are orthonormal, (iv) $v_1, \dots, v_r \in \mathbb{R}^n$ are orthonormal.



From the singular value decomposition one can see that $Av_i = \sigma_i u_i$ and $u_i^T A = \sigma_i v_i^T$ so one should think of the singular vectors u_i and v_i as a relaxed form of an eigenvector. As a rule of thumb, if one has an arbitrary matrix and needs to prove certain properties it is usually a good idea to study its singular value decomposition. We note that the singular value decomposition can be computed in time $O(n^3)$.

First we want to discuss the relationship of A to the matrices $A^T A$ and AA^T .

Lemma 4.18. Let $A \in \mathbb{R}^{m \times n}$ be a matrix with singular value decomposition $A = \sum_{i=1}^r \sigma_i u_i v_i^T$.

- (i) The matrix $AA^T \in \mathbb{R}^{m \times m}$ is symmetric and positive semidefinite with Eigenvectors u_1, \dots, u_r and positive Eigenvalues $\sigma_1^2, \dots, \sigma_r^2$. In particular $AA^T = \sum_{i=1}^r \sigma_i^2 u_i u_i^T$.
- (ii) The matrix $A^T A \in \mathbb{R}^{n \times n}$ is symmetric and positive semidefinite with Eigenvectors v_1, \dots, v_r and positive Eigenvalues $\sigma_1^2, \dots, \sigma_r^2$. In particular $A^T A = \sum_{i=1}^r \sigma_i^2 v_i v_i^T$.

Proof. We verify (i) as (ii) is similar. Replacing A and A^T with the SVD we can see that

$$AA^T = \left(\sum_{i=1}^r \sigma_i u_i v_i^T \right) \left(\sum_{j=1}^r \sigma_j u_j v_j^T \right)^T = \sum_{i=1}^r \sum_{j=1}^r \sigma_i \sigma_j u_i \underbrace{\langle v_i, v_j \rangle}_{\substack{=0 \text{ if } i \neq j, \\ =1 \text{ if } i=j}} u_j^T = \sum_{i=1}^r \sigma_i^2 u_i u_i^T$$

Since u_1, \dots, u_r are orthonormal, $\sigma_1^2, \dots, \sigma_r^2$ must be the Eigenvalues of AA^T . \square

4.3 Matrix norms

Recall that a *norm* on a (real) vector space V is a map $\|\cdot\| : V \rightarrow \mathbb{R}_{\geq 0}$ satisfying:

- (i) Triangle inequality: $\|x + y\| \leq \|x\| + \|y\|$ for all $x, y \in V$

(ii) Homogeneity: $\|sx\| = |s| \cdot \|x\|$ for all $x \in V$, $s \in \mathbb{R}$

(iii) Positivity: For all x , $\|x\| = 0 \Leftrightarrow x = \mathbf{0}$

We note that the above properties imply that $\|x\| \geq 0$ for all $x \in V$. Some textbooks include this explicitly but one can verify that indeed $0 = \|x - x\| \leq \|x\| + \|-x\| = 2\|x\|$ and so $\|x\| \geq 0$.

It is often useful to define norms for matrices as well (and not just the vector space \mathbb{R}^n). We mention the most popular ones for matrices $A \in \mathbb{R}^{m \times n}$:

- **The Frobenius norm.** We set

$$\|A\|_F := \sqrt{\text{Tr}[AA^T]} = \sqrt{\sum_{i=1}^n \sum_{j=1}^m A_{ij}^2}$$

The Frobenius norm corresponds to the Euclidean norm when interpreting A as a mn -dimensional vector. Moreover we can write

$$\|A\|_F = \sqrt{\text{Tr}[AA^T]} \stackrel{\text{Lem 4.10}}{=} \left(\sum_{i=1}^m \lambda_i(AA^T) \right)^{1/2} \stackrel{\text{Lem 4.18}}{=} \left(\sum_{i=1}^m \sigma_i(A)^2 \right)^{1/2}$$

using Lemma 4.10 and Lemma 4.18. That means $\|A\|_F$ is the Euclidean norm of the vector of singular values.

- **The Operator norm.** We define

$$\|A\|_{\text{op}} := \max_{x \in \mathbb{R}^n: \|x\|_2=1} \|Ax\|_2 = \max_{x \in \mathbb{R}^n \setminus \{\mathbf{0}\}} \frac{\|Ax\|_2}{\|x\|_2}$$

Loosely speaking this gives the maximum “stretch” of any vector. It turns out the operator norm corresponds to a value that we already know:

$$\begin{aligned} \|A\|_{\text{op}} &= \max_{\|x\|_2=1} \|Ax\|_2 = \left(\max_{\|x\|_2=1} \|Ax\|_2^2 \right)^{1/2} \\ &= \left(\max_{\|x\|_2=1} x^T A^T A x \right)^{1/2} \stackrel{\text{Thm 4.13}}{=} \sqrt{\lambda_1(A^T A)} \stackrel{\text{Lem 4.18}}{=} \sigma_1(A) \end{aligned}$$

4.4 Best low rank approximation

A recurrent algorithmic problem is the following: given a matrix $A \in \mathbb{R}^{m \times n}$, find the best low rank matrix B that approximates A well. If $A = \sum_{i=1}^r \sigma_i u_i v_i^T$ is the SVD of A , then a natural rank- k approximation (where $k \leq r$) would be $B =$

$\sum_{i=1}^k \sigma_i u_i v_i^T$. In other words, we simply truncate the singular value decomposition after the k largest singular values. We can rigorously prove that if we measure the approximation error in terms of the operator norm $\|\cdot\|_{\text{op}}$ or the Frobenius norm $\|\cdot\|_F$, then this choice is optimal. For both proofs, we fix the SVD $A = \sum_{i=1}^r \sigma_i u_i v_i^T$ with $\sigma_1 \geq \dots \geq \sigma_r > 0$ and assume $0 \leq k \leq r$.

Theorem 4.19. For any matrix $A \in \mathbb{R}^{m \times n}$ and any k one has

$$\inf_{B: \text{rank}(B)=k} \|A - B\|_{\text{op}} = \sigma_{k+1}$$

In class, we will discuss the first part of the proof and skip the 2nd part.

Proof. First we prove that for the choice of $B := \sum_{i=1}^k \sigma_i u_i v_i^T$ one has $\|A - B\|_{\text{op}} = \sigma_{k+1}$. And in fact

$$\|A - B\|_{\text{op}} = \left\| \sum_{i=k+1}^r \sigma_i u_i v_i^T \right\|_{\text{op}} \stackrel{\text{Sec 4.3}}{=} \sigma_{k+1}$$

Next, let $B \in \mathbb{R}^{m \times n}$ be an arbitrary matrix with $\text{rank}(B) = k$; we need to prove that $\|A - B\|_{\text{op}} \geq \sigma_{k+1}$. Let $U := \{x \in \mathbb{R}^n \mid Bx = \mathbf{0}\}$ be the nullspace of B . Note that $\dim(U) = n - k$. Then making use of the Raleigh coefficient twice we get

$$\begin{aligned} \|A - B\|_{\text{op}}^2 &= \lambda_{\max}((A - B)^T(A - B)) \\ &\stackrel{\text{Thm 4.13}}{=} \max_{\|x\|_2=1} x^T (A - B)^T (A - B) x \\ &\geq \max_{x \in U: \|x\|_2=1} x^T (A - B)^T (A - B) x \\ &\stackrel{Bx=0}{=} \max_{x \in U: \|x\|_2=1} x^T A^T A x \\ &\geq \min_{V: \dim(V)=n-k} \max_{x \in V: \|x\|_2=1} x^T (A^T A) x \stackrel{\text{Thm 4.16}}{=} \lambda_{k+1}(A^T A) = \sigma_{k+1}(A)^2 \end{aligned}$$

□

Theorem 4.20. For any matrix $A \in \mathbb{R}^{m \times n}$ and any k one has

$$\inf_{B: \text{rank}(B)=k} \|A - B\|_F^2 = \sum_{i=k+1}^r \sigma_i^2$$

We will not give the full proof in class and only list it for the sake of completeness.

Proof. Again, we first consider $B := \sum_{i=1}^k \sigma_i u_i v_i^T$. Then

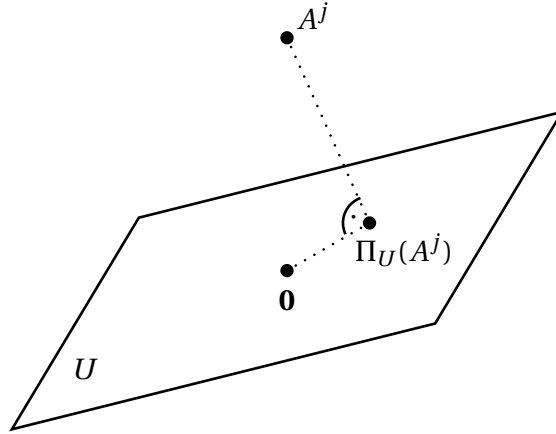
$$\|A - B\|_F^2 = \left\| \sum_{i=k+1}^r \sigma_i u_i v_i^T \right\|_F^2 \stackrel{\text{Phytagoras}}{=} \sum_{i=k+1}^r \|\sigma_i u_i v_i^T\|_F^2 = \sum_{i=k+1}^r \sigma_i^2$$

Here we use that the matrices $u_1 v_1^T, \dots, u_r v_r^T$ are orthogonal.

For the second part, we fix an arbitrary matrix $B \in \mathbb{R}^{m \times n}$ with $\text{rank}(B) = k$ and prove that $\|A - B\|_F^2 \geq \sum_{i=k+1}^r \sigma_i^2$. Let $U := \text{span}\{B^1, \dots, B^n\}$ be the span of the columns of B . Note that U is a k -dimensional subspace. The Frobenius distance of B to A is

$$\|A - B\|_F^2 = \sum_{j=1}^n \|A^j - B^j\|_2^2$$

We can make the following observation: after fixing the subspace U , the best choice for the column B^j so that $B^j \in U$ must be the orthogonal projection of A^j into U , i.e. w.l.o.g. $B_i = \Pi_U(A^j)$.



Let $w_1, \dots, w_k \in \mathbb{R}^m$ be an orthonormal basis of the subspace U . Consider the matrix $\Pi := \sum_{i=1}^k w_i w_i^T$. Then $\pi_U(x) = \sum_{i=1}^k \langle w_i, x \rangle w_i = \Pi x$. That means Π is the matrix representing the linear map Π_U . Note that Π is a PSD matrix that has Eigenvalues 0 and 1. Such matrices are called *projection matrices*. In particular those matrices have the property that $\Pi^2 = \Pi$. Continuing the argument from above we can write

$$\|A - B\|_F^2 = \sum_{j=1}^n \|A^j - \Pi A^j\|_2^2 = \|A - \Pi A\|_F^2$$

So it suffices to prove the following:

Claim I. Let $A \in \mathbb{R}^{m \times n}$. For any rank- k projection matrix Π one has $\|A - \Pi A\|_F^2 \geq \sum_{i \geq k+1} \sigma_i^2$ where $\sigma_1 \geq \sigma_2 \geq \dots$ are the singular values of A .

Proof of Claim I. Again we write w_1, \dots, w_k as the orthonormal set of vectors so that $\Pi = \sum_{\ell=1}^k w_\ell w_\ell^T$. We extend w_1, \dots, w_m to an orthonormal basis of \mathbb{R}^m and set $\Pi^\perp := \sum_{\ell=k+1}^m w_\ell w_\ell^T = I_m - \Pi$. Note that Π^\perp is the projection matrix for U^\perp (which is the $m - k$ dimensional subspace orthogonal to U). Note that $A - \Pi A = A(I_m - \Pi) = A\Pi^\perp$. Let $A = \sum_{i=1}^m \sigma_i u_i v_i^T$ be the SVD of A (which we “filled up” with singular values of value 0 in case that $\text{rank}(A)$ is less than m). Then

$$\begin{aligned}
\|A - \Pi A\|_F^2 &= \|A\Pi^\perp\|_F^2 \\
&= \text{Tr}[(A\Pi^\perp)^T(A\Pi^\perp)] \\
&= \text{Tr}[\Pi^\perp A^T A \Pi^\perp] \\
&\stackrel{\text{cyclicity}}{=} \text{Tr}[A^T A \underbrace{(\Pi^\perp)^2}_{=\Pi^\perp}] \\
&= \text{Tr}\left[\left(\sum_{i=1}^m \sigma_i^2 u_i u_i^T\right) \Pi^\perp\right] = \sum_{i=1}^m \sigma_i^2 \cdot u_i^T \Pi^\perp u_i \geq \sum_{i=k+1}^m \sigma_i^2
\end{aligned}$$

In the last step we use the following argument: we know that $u_i \Pi^\perp u_i \leq 1$ for all i and $\sum_{i=1}^m u_i^T \Pi^\perp u_i = \text{Tr}[\Pi^\perp] = m - k$. Hence the quantity in question must be at least as large as the sum of $m - k$ many σ_i^2 's. \square

Algorithmically it seems obvious that in order to compute the best rank- k approximation B to a matrix A , we first compute the SVD in time $O(n^3)$ and then truncate it to k summands. But it turns out that if one is satisfied with an approximate answer then computing the matrix B can be done directly and much faster.

Theorem 4.21 (Clarkson-Woodruff [CW13]). *Given a matrix $A \in \mathbb{R}^{n \times n}$ and parameters $\varepsilon > 0$ and $k \in \mathbb{N}$ one can find a rank- k matrix $B \in \mathbb{R}^{n \times n}$ so that*

$$\|A - B\|_F^2 \leq (1 + \varepsilon) \inf_{B^*: \text{rank}(B^*)=k} \|A - B^*\|_F^2$$

in time $O(\text{nnz}(A) \cdot (\frac{k}{\varepsilon} + k \ln(k))) + n \cdot \text{poly}(\frac{k}{\varepsilon})$.

Here $\text{nnz}(A)$ denotes the number of non-zero entries of the matrix A . For example if one thinks of ε and k as constants, then the algorithm takes $O(\text{nnz}(A) + n)$ which is *linear* in the input length,

4.5 Hidden Partition

We briefly discuss one elegant application of SVDs / Eigen decomposition. Suppose we have a *random graph* $G = ([n], E)$ which is generated as follows: there

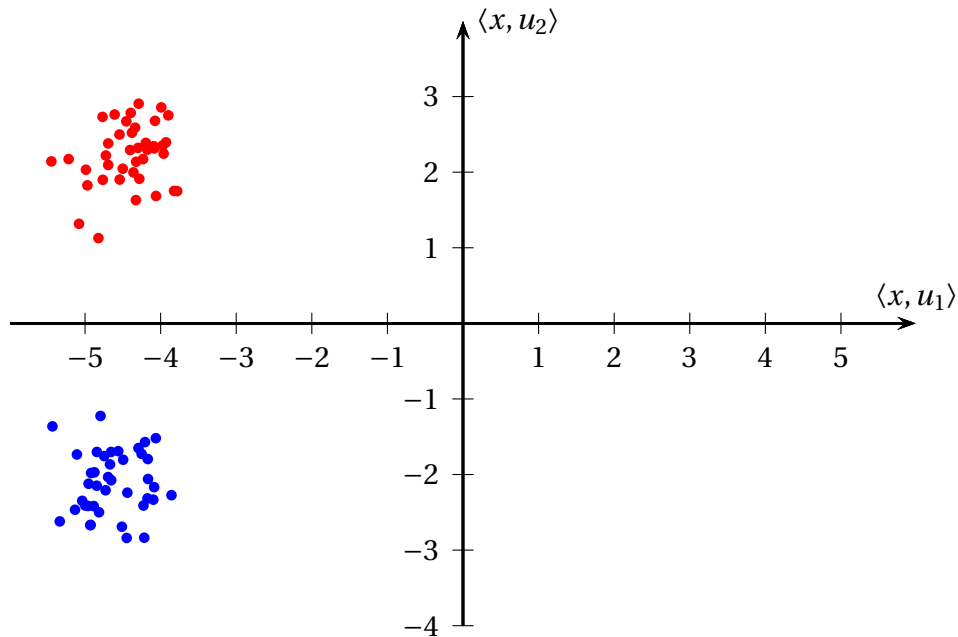
is an unknown *partition* $[n] = X \dot{\cup} Y$ of the vertices where $|X| = |Y| = \frac{n}{2}$. Let $1 > p > q > 0$ be constants. Then independently for all $i \neq j$ one has

$$\Pr[\{i, j\} \in E] = \begin{cases} p & \text{if } i, j \in X \text{ or } i, j \in Y \\ q & \text{if } |\{i, j\} \cap X| = 1 = |\{i, j\} \cap Y| = 1 \end{cases}$$

The goal is: given such a graph G , recover the partitions X and Y . One can think of this problem as discovering two unknown communities in the graph where an individuals in the same community are more likely connected than individuals in different communities. Note that each vertex i has an expected degree of $n \cdot \frac{p+q}{2}$, so one cannot directly use the degrees to infer what community i belongs to. Let $A \in \{0, 1\}^{n \times n}$ be the symmetric random matrix that is the adjacency matrix of G . We make the observation that in expectation the adjacency matrix is

$$\mathbb{E}[A] = \begin{array}{cc} & \begin{array}{cc} X & Y \end{array} \\ \begin{array}{c} p \cdot \mathbf{1}\mathbf{1}^T \\ \dots \\ q \cdot \mathbf{1}\mathbf{1}^T \end{array} & \begin{array}{c} q \cdot \mathbf{1}\mathbf{1}^T \\ \dots \\ p \cdot \mathbf{1}\mathbf{1}^T \end{array} \\ \begin{array}{c} X \\ Y \end{array} & \end{array}$$

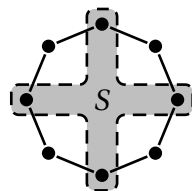
In particular $\text{rank}(\mathbb{E}[A]) = 2$. Of course, the actual outcome A may not be of rank 2, but with high probability, A would be close to the rank-2 matrix $\mathbb{E}[A]$. This suggests to look at the best rank-2 approximation of A . Let $A = \sum_{i=1}^n \lambda_i u_i u_i^T$ be the Eigen decomposition so that $|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_n| \geq 0$. Then from the earlier discussion we know that $B := \sum_{i=1}^2 \lambda_i u_i u_i^T$ is the rank-2 approximation to use. In fact, consider the map $\pi : \mathbb{R}^n \rightarrow \mathbb{R}^2$ with $\pi(x) := (\langle x, u_1 \rangle, \langle x, u_2 \rangle)$ that gives the coordinates of a projection into the span of B . Then consider the 2-dimensional points $\pi(A^1), \dots, \pi(A^n)$. Points of vertices in the same community will be closer than those in different communities. This can be used to exactly recover X and Y with high probability (assuming n is large enough depending on the distance $|p - q|$). For details see the works of McSherry [McS01] and Vu [Vu14].



Random hidden partition instance for $n = 40$, $p = \frac{3}{4}$, $q = \frac{1}{4}$ with points $\Pi_U(A^j)$ where U is the span of the 2 largest Eigenvectors of A .

4.6 Additive approximations for MaxCut

Consider an undirected graph $G = (V, E)$ with $n := |V|$ vertices. We have seen in Chapter 1 that one can find a *minimum cut* in polynomial time. Now we are interested in the opposite problem of finding a *maximum cut*, i.e. a set $\emptyset \subseteq S \subseteq V$ maximizing $|\delta(S)|$.



graph G with maxcut S

Somewhat surprisingly this problem turns out to be **NP-hard**. If we denote the optimum value by $OPT := \max\{|\delta(S)| : \emptyset \subseteq S \subseteq V\}$, then the seminal algorithm by Goemans and Williamson [GW95] finds a cut S in polynomial time so that $|\delta(S)| \geq 0.878 \cdot OPT$. One can also prove that if the so-called *Unique Games Conjecture* holds, then no polynomial time algorithm with a better approximation ratio exists.

In this section we want to give a polynomial time linear algebra-based algorithm to find a cut S that satisfies an *additive error* guarantee of $|\delta(S)| \geq OPT -$

εn^2 for any constant $\varepsilon > 0$. Such algorithms are only useful if the graph is *dense*, i.e. it has indeed $\Theta(n^2)$ many edges as otherwise the guarantee that can be reached in polynomial time is meaningless. But on the positive side, our method is very flexible and applies to many other problems. First we phrase the maxcut problem as a linear algebra question. Let $A \in \{0, 1\}^{n \times n}$ be the *adjacency matrix* of G where $A_{ij} = 1$ iff $\{i, j\} \in E$. We note that every adjacency matrix is symmetric. Then for any $S \subseteq [n]$ the value of the cut is

$$|\delta(S)| = \sum_{i \in S} \sum_{j \in \bar{S}} A_{ij} = \mathbf{1}_S^T A \mathbf{1}_{\bar{S}}$$

where $\bar{S} := [n] \setminus S$ is the complement of S and $\mathbf{1}_S$ is the characteristic vector of S . Then MaxCut can be phrased as the quadratic optimization problem

$$\max\{x^T A(\mathbf{1} - x) \mid x \in \{0, 1\}^n\} = \max\{(A\mathbf{1})^T x + x^T (-A)x \mid x \in \{0, 1\}^n\}$$

We prove that such quadratic optimization problems admit additive approximation algorithms.

Theorem 4.22. *Let $A \in [-1, 1]^{n \times n}$ be a symmetric matrix and let $c \in [-n, n]^n$. For any parameter $k \in \mathbb{N}$ one can approximate the problem*

$$\max\{c^T x + x^T A x \mid x \in \{0, 1\}^n\}$$

with an additive $O(\frac{n^2}{\sqrt{k}})$ error in time $n^{O(k)}$.

For the specific problem of MaxCut one can reduce the running time to $k^k \cdot \text{poly}(n)$ [OGT15] but here we prefer generality and simplicity over efficiency. The first step towards Theorem 4.22 is to approximate A by a low rank matrix B and prove that the new matrix does not change the objective function by much.

Lemma 4.23. *Let $B \in \mathbb{R}^{n \times n}$ be the best rank- k approximation to A as in Theorem 4.19. Then for all $x \in \{0, 1\}^n$*

$$|x^T A x - x^T B x| \leq n \cdot \sigma_{k+1}(A) \leq \frac{n^2}{\sqrt{k+1}}.$$

Proof. Since A is symmetric we can work with the slightly simpler Eigen decomposition $A = \sum_{i=1}^n \lambda_i v_i v_i^T$ instead of the SVD. We sort the Eigenvalues so that $|\lambda_1| \geq \dots \geq |\lambda_n|$ which are the singular values of A in non-increasing order. Then

$$|x^T (A - B)x| \stackrel{(*)}{\leq} \|x\|_2 \cdot \|(A - B)x\|_2 \leq \underbrace{\|x\|_2}_{\leq \sqrt{n}} \cdot \underbrace{\|A - B\|_{\text{op}}}_{\substack{\leq |\lambda_{k+1}| \\ \text{by Thm 4.19}}} \cdot \underbrace{\|x\|_2}_{\leq \sqrt{n}} \leq n \cdot |\lambda_{k+1}|$$

where we use Cauchy-Schwarz³ in (*) and Theorem 4.19 in the last step⁴. It remains to prove that the $(k+1)$ st singular value $|\lambda_{k+1}|$ is small. And indeed because $|\lambda_1| \geq \dots \geq |\lambda_n|$ we have that

$$\lambda_{k+1}^2 \leq \frac{\lambda_1^2 + \dots + \lambda_{k+1}^2}{k+1} \leq \frac{\sum_{i=1}^n \lambda_i^2}{k+1} = \frac{\|A\|_F^2}{k+1} \leq \frac{n^2}{k+1}$$

Here we use the insight from Sec 4.3 that $\|A\|_F^2$ is the squared sum of singular values of A together with the observation that for any matrix $A \in [-1, 1]^{n \times n}$ one has $\|A\|_F^2 \leq n^2$. Putting everything together we get

$$|x^T (A - B)x| \leq n|\lambda_{k+1}| \leq n \cdot \sqrt{\frac{n^2}{k+1}}$$

which is the claimed bound. \square

Now we describe the remainder of the proof of Theorem 4.22.

Proof of Theorem 4.22. As we have seen in Lemma 4.23, after replacing the matrix A by the rank- k approximation B it suffices to approximate the problem

$$\max \{c^T x + x^T Bx \mid x \in \{0, 1\}^n\}$$

where B has rank k . Note that it is not necessarily true anymore that $|B_{ij}| \leq 1$ for all i, j , but we still know that $\|B\|_F \leq n$. Let $B = \sum_{i=1}^k \lambda_i v_i v_i^T$ be the Eigen decomposition of B . Then the objective function is

$$c^T x + x^T Bx = c^T x + \sum_{i=1}^k \lambda_i \langle v_i, x \rangle^2$$

In order to make the problem even simpler we *discretize* the vectors. For a parameter $\delta > 0$ that we determine later, we round all entries in v_i to the nearest multiple of δ that has a smaller absolute value and denote the new vector by \tilde{v}_i . We do the same and replace c by \tilde{c} . Then we set $\tilde{B} := \sum_{i=1}^k \lambda_i \tilde{v}_i \tilde{v}_i^T$. We can prove that also this discretization does not incur too much of an error:

Claim I. For all $x \in \{0, 1\}^n$ one has $|(c^T x + x^T Bx) - (\tilde{c}^T x + x^T \tilde{B}x)| \leq O(\sqrt{kn}^{5/2} \delta)$.

Proof of Claim I. We bound the error of the linear part and the quadratic part separately. For the linear part we have

$$|c^T x - \tilde{c}^T x| \leq \underbrace{\|c - \tilde{c}\|_\infty}_{\leq \delta} \underbrace{\|x\|_1}_{\leq n} \leq \delta n$$

³Which says that for any vectors a, b one has $|\langle a, b \rangle| \leq \|a\|_2 \cdot \|b\|_2$.

⁴To be very precise, the SVD of A is $A = \sum_{i=1}^n \sigma_i u_i v_i^T$ with $\sigma_i := |\lambda_i|$ and $u_i := \text{sign}(\lambda_i) v_i$.

Next, we consider the quadratic part:

$$\begin{aligned}
|x^T Bx - x^T \tilde{B}x| &= \left| \sum_{i=1}^k \lambda_i (\langle v_i, x \rangle^2 - \langle \tilde{v}_i, x \rangle^2) \right| \\
&\leq \sum_{i=1}^k |\lambda_i| \cdot |\langle v_i, x \rangle + \langle \tilde{v}_i, x \rangle| \cdot |\langle v_i - \tilde{v}_i, x \rangle| \\
&\leq \sum_{i=1}^k |\lambda_i| \cdot \underbrace{(\|v_i\|_2 + \|\tilde{v}_i\|_2)}_{\leq 1} \cdot \underbrace{\|x\|_2}_{\leq \sqrt{n}} \cdot \underbrace{\|v_i - \tilde{v}_i\|_\infty}_{\leq \delta} \underbrace{\|x\|_1}_{\leq n} \\
&\leq \underbrace{\sqrt{k} \left(\sum_{i=1}^k \lambda_i^2 \right)^{1/2}}_{=\|B\|_F \leq n} \cdot n^{3/2} \delta \\
&\leq \sqrt{k} n^{5/2} \delta
\end{aligned}$$

Here we use the binomial equation $a^2 - b^2 = (a + b)(a - b)$ for numbers $a, b \in \mathbb{R}$, Cauchy-Schwarz in the form $|\langle a, b \rangle| \leq \|a\|_2 \|b\|_2$ for vectors a, b as well as $|\langle a, b \rangle| \leq \|a\|_\infty \|b\|_1$. Moreover we have used $\|a\|_1 \leq \sqrt{k} \|a\|_2$ for any k -dimensional vector. \square

After the low rank approximation and the discretization the problem that remains to be solved is

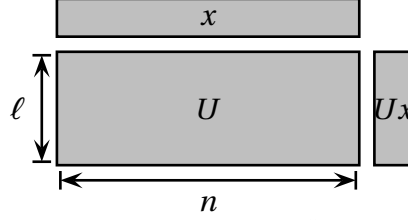
$$\max \left\{ \tilde{c}^T x + \sum_{i=1}^k \lambda_i \langle \tilde{v}_i, x \rangle^2 : x \in \{0, 1\}^n \right\} \quad (4.1)$$

From the analysis we have seen that setting for example $\delta := \frac{1}{\sqrt{kn}}$ would suffice. But note that the number of solutions x is still 2^n so it is still not obvious how to solve (4.1) time $n^{O(k)}$. For $x \in \{0, 1\}^n$ consider the $(k+1)$ -dimensional vector

$$\phi(x) := \begin{pmatrix} \tilde{c}^T x \\ \tilde{v}_1^T x \\ \vdots \\ \tilde{v}_k^T x \end{pmatrix}$$

We claim that there are at most $(n/\delta)^{O(k)}$ many vectors $\phi(x)$ and they can be computed in time $(n/\delta)^{O(k)}$ as well. We phrase this fact in a self-contained way:

Claim II. Let $U \in \{-\Delta, \dots, \Delta\}^{\ell \times n}$ where $\Delta \in \mathbb{N}$ and let $\mathcal{F} := \{Ux \mid x \in \{0, 1\}^n\}$. Then $|\mathcal{F}| \leq (2\Delta n + 1)^\ell$. Moreover \mathcal{F} can be computed in time $(\Delta n)^{O(\ell)}$.



Proof of Claim II. First note that for $x \in \{0, 1\}^n$, Ux is an ℓ -dimensional integer vector with entries of size $\|Ux\|_\infty \leq n\Delta$. Then $|\mathcal{F}| \leq | \{-\Delta n, \dots, \Delta n\}^\ell | = (2\Delta n + 1)^\ell$. We can compute \mathcal{F} using *dynamic programming*. We use the table entries

$$T(b, j) := \begin{cases} \text{TRUE} & \text{if } \exists x \in \{0, 1\}^j : \sum_{i=1}^j U^i x_i = b \\ \text{FALSE} & \text{otherwise} \end{cases}$$

for $j = 0, \dots, n$ and $b \in \mathbb{Z}^\ell$ with $\|b\|_\infty \leq \Delta n$. Note that $T(b, j) = 1$ means that the vector b can be generated using the first j columns of U . Then using the Bellman equation

$$T(b, j) := T(b, j-1) \vee T(b - U^j, j-1)$$

we can compute all entries. The running time is dominated by the number of entries which is $(n+1) \cdot (2\Delta n + 1)^\ell$. \square

In order to solve (4.1) we apply Claim II for the matrix U with $\ell = k+1$ rows of the form $\frac{\tilde{c}}{\delta}, \frac{\tilde{v}_1}{\delta}, \dots, \frac{\tilde{v}_k}{\delta}$. Given a vector $\phi(x)$ we can compute the objective $\tilde{c}^T x + \sum_{i=1}^k \lambda_i \langle \tilde{v}_i, x \rangle^2$ without knowing x itself. \square

We would like to mention that we could choose δ much smaller without asymptotically affecting the running time and so really the error is dominated by the truncation to rank k and not the discretization. For example one could restate the result as follows:

Corollary 4.24. *Let $A \in [-1, 1]^{n \times n}$ be a symmetric matrix and let $c \in [-n, n]^n$. For any parameter $k \in \mathbb{N}$ one can approximate the problem*

$$\max \{ c^T x + x^T A x \mid x \in \{0, 1\}^n \}$$

with an additive $n \cdot \sigma_{k+1}(A) + \frac{1}{100n}$ error in time $n^{O(k)}$.

Hence if the matrix A was already “mostly” low rank to begin with then the approximation error might have been pretty good.

Chapter 5

Spectral graph theory

Many algorithmic problems deal with (undirected) *graphs* and in this chapter we will discuss how to understand graphs using a linear-algebraic perspective. A *weighted undirected graph* is of the form $G = (V, E, w)$ where $w : E \rightarrow \mathbb{R}_{>0}$ are positive edge weights. For a subset $F \subseteq E$ of edges we use the notation $w(F) := \sum_{e \in F} w_e$. Recall that for $S \subseteq V$, $\delta(S) := \{e \in E \mid |e \cap S| = 1\}$ are the edges crossing the cut. In particular we use $w(\delta(i)) = \sum_{j: \{i,j\} \in E} w_{ij}$ to denote the weighted degree of a node $i \in V$.

Definition 5.1. Given a weighted undirected graph $G = (V, E, w)$, the *weighted adjacency matrix* is the symmetric matrix $A \in \mathbb{R}^{V \times V}$ with entries

$$A_{ij} := A_{ji} := \begin{cases} w_{ij} & \text{if } \{i, j\} \in E \\ 0 & \text{otherwise} \end{cases}$$

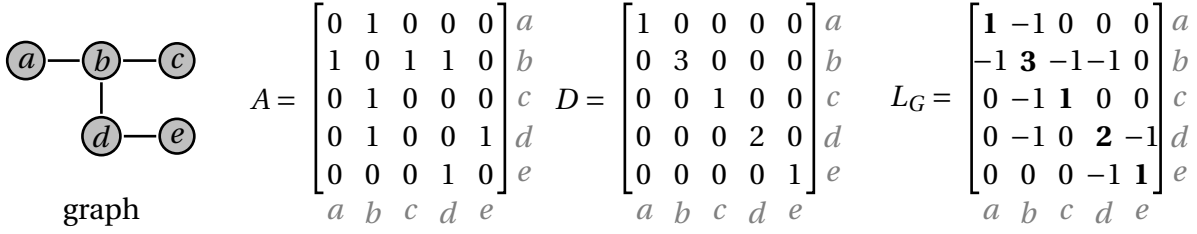
The *degree matrix* is the diagonal matrix $D \in \mathbb{R}^{V \times V}$ with the weighted degrees on the diagonal, i.e.

$$D_{ii} := w(\delta(i)) \quad \forall i \in V$$

The (*weighted*) *Laplacian* $L_G \in \mathbb{R}^{V \times V}$ is the symmetric matrix defined by $L_G := D - A$. In other words,

$$(L_G)_{ij} := \begin{cases} w(\delta(i)) & \text{if } i = j \\ -w_{ij} & \text{if } \{i, j\} \in E \\ 0 & \text{otherwise} \end{cases}$$

We depict an example with unit edge weights below:



$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix} \quad D = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix} \quad L_G = \begin{bmatrix} \mathbf{1} & -1 & 0 & 0 & 0 \\ -1 & \mathbf{3} & -1 & -1 & 0 \\ 0 & -1 & \mathbf{1} & 0 & 0 \\ 0 & -1 & 0 & \mathbf{2} & -1 \\ 0 & 0 & 0 & -1 & \mathbf{1} \end{bmatrix} \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix}$$

graph

adjacency matrix

degree matrix

Laplacian

We claim that many properties of the graph can be derived from the Laplacian L_G . Note that for all $x \in \mathbb{R}^V$ one has

$$x^T L_G x = x^T (D - A)x = \sum_{i \in V} w(\delta(i)) \cdot x_i^2 - 2 \sum_{\{i,j\} \in E} w_{ij} x_i x_j = \sum_{\{i,j\} \in E} w_{ij} (x_i - x_j)^2 \geq 0$$

The map $x \mapsto x^T L_G x$ is also called the *quadratic form* of L_G .

Lemma 5.2. *For any weighted graph G , the Laplacian L_G is positive semidefinite. In fact, $0 \leq L_G \leq 2D$.*

Proof. The first claim follows from Lemma 4.5 as $x^T L_G x \geq 0$ for all $x \in \mathbb{R}^V$. Now we prove that $L_G \leq 2D$. It will be useful to note that for all $a, b \in \mathbb{R}$ one has $(a - b)^2 \leq 2a^2 + 2b^2$. Then for each $x \in \mathbb{R}^V$ one has

$$x^T L_G x = \sum_{\{i,j\} \in E} w_{ij} (x_i - x_j)^2 \leq \sum_{\{i,j\} \in E} w_{ij} \cdot (2x_i^2 + 2x_j^2) = 2x^T D x.$$

□

Alternatively we note that a Laplacian can be written as a sum of (PSD) rank-1 matrices

$$L_G = \sum_{\{i,j\} \in E} w_{ij} (e_i - e_j)(e_i - e_j)^T$$

and hence must be PSD.

Fact 5.3. For any weighted graph G with $n = |V|$ one has $\lambda_{\min}(L_G) = 0$. In fact, $\mathbf{1}$ is an Eigenvector with Eigenvalue 0.

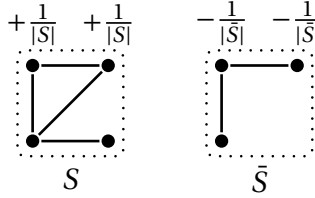
Proof. For any $i \in V$ one has $\langle (L_G)_i, \mathbf{1} \rangle = w(\delta(i)) - w(\delta(i)) = 0$ which means that $L_G \mathbf{1} = 0 \cdot \mathbf{1}$. □

In the following we denote $\lambda_i := \lambda_i(L_G)$ as the i th Eigenvalue of L_G and sort the indices so that¹ $0 \leq \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$.

¹Note that previously we used the reverse ordering. But in spectral graph theory it is standard to denote λ_1 as the smallest and not the largest Eigenvalue, so we follow the convention used in the literature.

Lemma 5.4. For any weighted graph G one has $\lambda_2 = 0 \Leftrightarrow G$ is disconnected.

Proof. (\Leftarrow). We assume that G is disconnected and need to prove that there is an Eigenvector orthogonal to $\mathbf{1}$ that also has Eigenvalue 0. First, there is a cut $\emptyset \subset S \subset V$ with $\delta(S) = \emptyset$. Let $x \in \mathbb{R}^V$ be the vector with $x := \frac{\mathbf{1}_S}{|S|} - \frac{\mathbf{1}_{\bar{S}}}{|\bar{S}|}$.



Then clearly $\langle \mathbf{1}, x \rangle = \frac{|S|}{|S|} - \frac{|\bar{S}|}{|\bar{S}|} = 0$. Moreover for any $\{i, j\} \in E$ one has $x_i = x_j$ and so $x^T L_G x = \sum_{\{i,j\} \in E} w_{ij} (x_i - x_j)^2 = 0$. Hence x is indeed an Eigenvector with Eigenvalue 0.

(\Rightarrow). Suppose for the sake of contradiction that G is connected and $\lambda_2 = 0$. Then there is an Eigenvector x with Eigenvalue 0 so that $\mathbf{1} \perp x$. As $\sum_{i \in V} x_i = 0$, we know that x is not constant. As G is connected, there must be neighboring vertices i^*, j^* so that $x_{i^*} \neq x_{j^*}$. Then

$$0 = x^T \underbrace{L_G x}_{=0} = \sum_{\{i,j\} \in E} \underbrace{w_{ij} (x_i - x_j)^2}_{>0 \text{ for } i=i^*, j=j^*} > 0$$

This is a contradiction. □

Interestingly, the second smallest Eigenvalue $\lambda_2(L_G)$ not only determines whether G is connected, but also how strongly G is connected. This is one of the most important results in spectral graph theory and we discuss it in the next section.

5.1 Graph partitioning

Given a graph G , a frequent algorithmic problem is to *partition* the graph into two disjoint sets S and $V \setminus S$ so that few edges are separated. We have seen the polynomial-time solvable MinCut problem that minimizes the number (or weight) of separated edges in Chapter 1. But often it is not only the weight of the separated edges that should be small but one also would like that the sets S and $[n] \setminus S$ are both large. For example if the partition is the basis for a recursive algorithm it is less effective if one of the sets is a singleton. So we want to introduce a different objective function that takes the size of the sets into account.

Definition 5.5. For a weighted graph $G = (V, E, w)$ and vertices $S \subseteq V$ we define the *volume* as

$$\text{vol}(S) := \sum_{i \in S} w(\delta(i))$$

which is the weighted sum of degrees of vertices in S . The *conductance* of S is

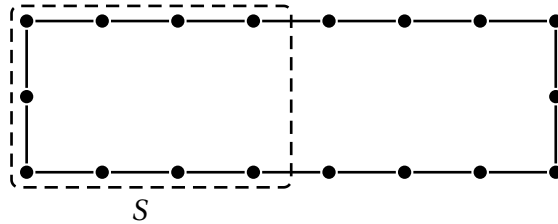
$$\phi(S) := \frac{w(\delta(S))}{\text{vol}(S)}$$

We note that $\phi(S)$ is the fraction of all the edge weight incident to S that is separated by S . By definition we have $0 \leq \phi(S) \leq 1$. We are interesting in finding a set S that minimizes the conductance. Note that $w(\delta(S)) = w(\delta(V \setminus S))$ and so we want to use the smaller of both sets S and $V \setminus S$ to count.

Definition 5.6. For a weighted graph $G = (V, E, w)$ we define the *conductance* of the graph itself as

$$\phi(G) := \min \left\{ \phi(S) \mid S \subseteq V \text{ with } \text{vol}(S) \leq \frac{1}{2} \text{vol}(V) \right\}$$

For example if G is a cycle of length n (with n even) and all edges have unit weight, then $\phi(G) = \frac{2}{n}$. So the conductance indeed gives a good definition for how good a graph can be divided.



cycle with cut S attaining conductance

In contrast to finding a minimum cut in a graph, determining the conductance is **NP-hard**. Thus we will discuss how $\phi(G)$ can be approximated.

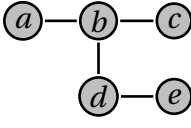
5.2 Cheeger's Inequality

We allow that vertices in our graph have different (weighted) degrees. First we need to introduce how to rescale the Laplacian to take that into account.

Definition 5.7. Let $G = (V, E, w)$ be a weighted graph on n vertices with weighted adjacency matrix A and weighted degree matrix D . Then the *normalized Laplacian* of G is the matrix $\tilde{L}_G \in \mathbb{R}^{V \times V}$ with

$$\tilde{L}_G := D^{-1/2} L_G D^{-1/2} = I_n - D^{-1/2} A D^{-1/2}$$

In the example graph from earlier, the normalized Laplacian looks as follows:



graph

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix}$$

degree matrix

$$\tilde{L}_G = \begin{bmatrix} 1 & -\frac{1}{\sqrt{3}} & 0 & 0 & 0 \\ \frac{1}{\sqrt{3}} & 1 & -\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{6}} & 0 \\ 0 & -\frac{1}{\sqrt{3}} & 1 & 0 & 0 \\ 0 & -\frac{1}{\sqrt{6}} & 0 & 1 & -\frac{1}{\sqrt{2}} \\ 0 & 0 & 0 & -\frac{1}{\sqrt{2}} & 1 \end{bmatrix} \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix}$$

normalized Laplacian

We note that \tilde{L}_G is rescaled so that the diagonal entries are all 1. From Lemma 5.2 we can derive that $0 \leq \tilde{L}_G \leq 2I_n$, i.e. all Eigenvalues of the normalized Laplacian lie in the interval $[0, 2]$. But it is not true anymore that all row sums and column sums are 0. Instead $D^{1/2}\mathbf{1}$ is an Eigenvector with Eigenvalue 0^2 . A bit informally speaking one can think of \tilde{L}_G as a rescaling so that all vertices are equally important. An important insight is that the Eigenvector belonging to $\lambda_2(\tilde{L}_G)$ can be used to extract a set of low conductance. For this we use the following algorithm:

SPECTRAL PARTITIONING ALGORITHM

Input: Weighted graph $G = (V, E, w)$.

Output: Low conductance cut $S \subseteq V$.

- (1) Compute the Eigenvector $x \in \mathbb{R}^V$ belonging to Eigenvalue $\lambda_2 := \lambda_2(\tilde{L}_G)$.
- (2) Sort (and rename) vertices so that

$$\frac{x_1}{\sqrt{d_1}} \leq \frac{x_2}{\sqrt{d_2}} \leq \dots \leq \frac{x_n}{\sqrt{d_n}}$$

where $d_i := w(\delta(i))$ is the weighted degree of node i .

- (3) Return the set $S := \{1, \dots, i\}$ attaining

$$\min \left\{ \phi(\{1, \dots, i\}) \mid \text{vol}(\{1, \dots, i\}) \leq \frac{\text{vol}(G)}{2}, i \in [n] \right\}$$

The algorithm will be used to prove the following fundamental result:

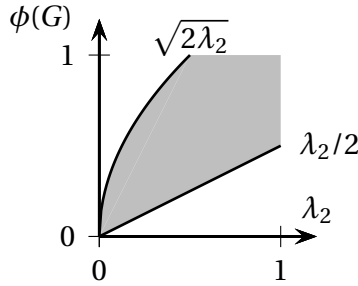
²One can check that indeed $\tilde{L}_G D^{1/2} \mathbf{1} = I_n D^{1/2} \mathbf{1} - D^{-1/2} A D^{-1/2} D^{1/2} \mathbf{1} = D^{1/2} \mathbf{1} - D^{-1/2} A \mathbf{1} = D^{-1/2} \mathbf{1} - D^{-1/2} D \mathbf{1} = \mathbf{0}$

Theorem 5.8 (Cheeger's Inequality). *For any weighted graph $G = (V, E, w)$ one has*

$$\frac{\lambda_2}{2} \leq \phi(G) \leq \sqrt{2\lambda_2}$$

where $\lambda_2 := \lambda_2(\tilde{L}_G)$.

The usefulness of this result lies in the fact that while computing $\phi(G)$ is **NP**-hard, the quantity λ_2 can be computed in polynomial time and so the spectral partitioning algorithm runs in polynomial time. It is a bit unfortunate that λ_2 and $\phi(G)$ are not within some fixed factors of each other, but Cheeger's inequality is tight and neither inequality can be improved.



possible combinations of $(\lambda_2, \phi(G))$

There are two alternative methods to approximate the conductance $\phi(G)$. Using *linear programming* one can find a cut S in polynomial time so that $\phi(S) \leq O(\log n) \cdot \phi(G)$ [LR99]. Moreover using *semidefinite programming* one can compute a cut S so that $\phi(S) \leq O(\sqrt{\log(n)}) \cdot \phi(G)$ [ARV09]. We now give the proof of Cheeger's inequality where for the second part we follow the notes of Lap Chi Lau³.

Proof of Cheeger's Inequality. We abbreviate $\mathcal{R}(x) := \frac{x^T \tilde{L}_G x}{\|x\|_2^2}$, which is the Raleigh coefficient for a vector $x \in \mathbb{R}^V$. We recall that $D^{1/2} \mathbf{1}$ is the Eigenvector of \tilde{L}_G with Eigenvalue 0. By Theorem 4.16 we then know that

$$\lambda_2 = \min \{ \mathcal{R}(x) : x \perp D^{1/2} \mathbf{1} \}$$

First we prove the easy direction which is that $\frac{\lambda_2}{2} \leq \phi(G)$. So, let S be the cut minimizing the conductance. We want to use S to define a vector orthogonal to $D^{1/2} \mathbf{1}$ so that its Raleigh coefficient is at most $2\phi(S)$. For this purpose we set $y := \mathbf{1}_S - \frac{\text{vol}(S)}{\text{vol}(V)} \mathbf{1}$ and $x := D^{1/2} y$. We claim that x certifies that $\lambda_2 \leq \mathcal{R}(x) \leq 2\phi(S)$.

³See <https://cs.uwaterloo.ca/~lapchi/cs860/notes/04-Cheeger.pdf>

Claim I. One has $x \perp D^{1/2}\mathbf{1}$ and $\mathcal{R}(x) \leq 2\phi(S)$.

Proof of Claim I. We verify that that indeed

$$\langle x, D^{1/2}\mathbf{1} \rangle = \langle y, D\mathbf{1} \rangle = \underbrace{\langle \mathbf{1}_S, D\mathbf{1} \rangle}_{=\text{vol}(S)} - \frac{\text{vol}(S)}{\text{vol}(V)} \underbrace{\langle \mathbf{1}, D\mathbf{1} \rangle}_{=\text{vol}(V)} = 0$$

Next, we estimate the nominator and denominator of the Raleigh coefficient $\mathcal{R}(x)$. First,

$$x^T \tilde{L}_G x \stackrel{\text{Def } y \text{ and } \tilde{L}_G}{=} y^T \underbrace{D^{1/2}(D^{-1/2}L_G D^{-1/2})}_{=I_n} D^{1/2} y = y^T L_G y \stackrel{L_G \mathbf{1} = \mathbf{0}}{=} \mathbf{1}_S^T L_G \mathbf{1}_S = w(\delta(S))$$

Secondly

$$\|x\|_2^2 = y^T D y = \langle y, D\mathbf{1}_S \rangle - \frac{\text{vol}(S)}{\text{vol}(V)} \underbrace{\langle y, D\mathbf{1} \rangle}_{=0} = \underbrace{\mathbf{1}_S^T D \mathbf{1}_S}_{=\text{vol}(S)} - \underbrace{\frac{\text{vol}(S)}{\text{vol}(V)} \mathbf{1}^T D \mathbf{1}_S}_{\leq 1/2} \geq \frac{1}{2} \text{vol}(S)$$

Finally

$$\mathcal{R}(x) = \frac{x^T \tilde{L}_G x}{x^T x} \leq \frac{w(\delta(S))}{\frac{1}{2} \text{vol}(S)} = 2\phi(S)$$

In the remainder, we prove the substantially harder direction of $\phi(G) \leq \sqrt{2\lambda_2}$. In order to ease notation we only give the proof for the case that G is d -regular (i.e. $|\delta(i)| = d$ for all $i \in V$) and the edges have unit weight. With this simplification we have

$$\tilde{L}_G = \frac{1}{d} L_G \quad \text{and} \quad \phi(G) = \min \left\{ \frac{|\delta(S)|}{d|S|} : |S| \leq \frac{n}{2} \right\}$$

Let x be the Eigenvector w.r.t. Eigenvalue λ_2 , i.e. $x \perp \mathbf{1}$ and $\mathcal{R}(x) = \lambda_2$. The strategy is to use x to extract a cut S of low conductance. That would be easy if say $x_i \in \{-1, +1\}$ for all i — but that may not be the case and we have to work a bit harder. The first step is to replace x by a vector that is easier to use for rounding to a cut.

We define $x^+ := \max\{x, \mathbf{0}\}$ as the vector where all negative entries have been replaced by 0's. Similarly $x^- := \min\{x, \mathbf{0}\}$ is the vector where all positive entries have been replaced by zeroes. Note that $x = x^+ + x^-$. We can prove that both vectors x^+ and x^- must have a Raleigh coefficient that is not larger than x — just that x^+ and x^- are not orthogonal to $\mathbf{1}$ anymore.

Claim II. One has $\mathcal{R}(x^+), \mathcal{R}(x^-) \leq \mathcal{R}(x) \leq \lambda_2$.

Proof of Claim II. By symmetry it suffices to prove that $\mathcal{R}(x^+) \leq \lambda_2$. In fact, the Raleigh coefficient is of the form

$$\mathcal{R}(x^+) = \frac{\sum_{i \in \text{supp}(x^+)} x_i^+ \langle (\tilde{L}_G)_i, x^+ \rangle}{\sum_{i \in \text{supp}(x^+)} x_i^+ \cdot x_i^+} \leq \max_{i \in \text{supp}(x^+)} \frac{\langle (\tilde{L}_G)_i, x^+ \rangle}{x_i^+}$$

Hence it suffices to prove that for every index $i \in \text{supp}(x^+)$ one has $\langle (\tilde{L}_G)_i, x^+ \rangle \leq \lambda_2 x_i^+$. And indeed

$$\langle (\tilde{L}_G)_i, x^+ \rangle = \underbrace{x_i^+}_{=x_i} - \frac{1}{d} x^+(\delta(i)) \stackrel{x^+ \geq x}{\leq} x_i - x(\delta(i)) = \langle (\tilde{L}_G)_i, x \rangle \stackrel{(*)}{=} \lambda_2 x_i = \lambda_2 x_i^+$$

where we crucially use in (*) that x is an Eigenvector and so $\tilde{L}_G x = \lambda_2 x$. \square

Now by picking the choice of $y \in \{\frac{x^+}{\|x^+\|_\infty}, -\frac{x^-}{\|x^-\|_\infty}\}$ with the smaller support we obtain a vector satisfying (i) $y \geq \mathbf{0}$, (ii) $1 \leq |\text{supp}(y)| \leq \frac{n}{2}$, (iii) $\mathcal{R}(y) \leq \lambda_2$, (iv) $\|y\|_\infty = 1$. We will use this vector y to generate a cut S . Now we come to the actual rounding step.

Claim III. *There is a threshold $0 < t \leq 1$ so that $S_t := \{i \in V \mid y_i^2 \geq t\}$ is non-empty and $\phi(S_t) \leq \sqrt{2\mathcal{R}(y)}$.*

Proof of Claim III. We draw $t \sim (0, 1]$ uniformly at random and study how good of a choice the random cut S_t is. First we bound the expected number of edges that are separated:

$$\begin{aligned} \mathbb{E}[|\delta(S_t)|] &= \sum_{\{i,j\} \in E} \underbrace{\Pr[\{i,j\} \in S_t]}_{=|y_i^2 - y_j^2|} \\ &= \sum_{\{i,j\} \in E} |y_i - y_j| \cdot |y_i + y_j| \\ &\stackrel{\text{Cauchy-Schwarz}}{\leq} \sqrt{\sum_{\{i,j\} \in E} |y_i - y_j|^2} \cdot \sqrt{\sum_{\{i,j\} \in E} |y_i + y_j|^2} \\ &\stackrel{(a+b)^2 \leq 2a^2 + 2b^2}{\leq} \sqrt{y^T L_G y} \cdot \sqrt{2d \sum_{i \in V} y_i^2} \end{aligned}$$

Next, we consider the average number of nodes in the cut:

$$\mathbb{E}[|S_t|] = \sum_{i \in V} \Pr[i \in S_t] = \sum_{i \in V} y_i^2$$

Then the ratio of those averages is

$$\frac{\mathbb{E}[|\delta(S_t)|]}{\mathbb{E}[d|S_t|]} \leq \frac{\sqrt{y^T L_G y} \cdot \sqrt{2d \sum_{i \in V} y_i^2}}{d \sum_{i \in V} y_i^2} = \sqrt{\frac{2 y^T \tilde{L}_G y}{\|y\|_2^2}} = \sqrt{2\mathcal{R}(y)}$$

Then there has to be *some* outcome of t where this inequality holds⁴. \square

Finally, inspecting Claim II and Claim III we note that the cut S_t that is produced is indeed of the form $S = \{1, \dots, i\}$ for some i , assuming we have sorted the vertices so that $x_1 \leq x_2 \leq \dots \leq x_n$ (for regular graphs). Hence (at least for regular graphs) this confirms that the cut S produced by the spectral rounding algorithm satisfies the claimed guarantee of $\phi(S) \leq \sqrt{2\lambda_2}$. \square

5.3 The power method

In many algorithms that depend on linear algebra it is more important that the linear algebra part works fast than it is that the answer is exact. For example computing the singular value decomposition (SVD) takes time $O(n^3)$ for an $n \times n$ matrix, which often is prohibitively slow. We want to show case one example where a linear algebraic quantity can be approximated with a very fast algorithm. Suppose $M \in \mathbb{R}^{n \times n}$ is a positive semidefinite matrix with Eigenvalues $\lambda_i := \lambda_i(M)$ sorted so that $\lambda_1 \geq \dots \geq \lambda_n \geq 0$. We are interested in approximately computing λ_1 and the corresponding Eigenvector. Consider the following randomized algorithm:

POWER METHOD

Input: Matrix $M \in \mathbb{R}^{n \times n}$ with $M \geq 0$ and parameter k

Output: Approximate Eigenvector x and Eigenvalue.

- (1) Choose a random Gaussian $x \in \mathbb{R}^n$, i.e. $x_1, \dots, x_n \sim \mathcal{N}(0, 1)$
- (2) FOR $i = 1$ TO k DO
 - (3) Update $x' := Mx$
- (4) Return x and $\frac{x^T Mx}{\|x\|_2^2}$

In terms of running times, the algorithm only needs k many matrix-vector multiplications which are particularly effective in the important case where M is *sparse*. Hence, let $\text{nnz}(M)$ denote the number of non-zero entries in M . Then one matrix-vector multiplication takes time $O(\text{nnz}(M))$ hence the total running time is $O(k \cdot \text{nnz}(M))$. Before we come to the formal analysis, we want to give an intuitive explanation why the algorithm works. Any PSD matrix has an Eigen

⁴This claim takes a bit more care than it seems. Suppose we have a distribution \mathcal{D} over some parameter t and two positive functions $f(t), g(t)$ that depend on that parameter. Say we know that $\frac{\mathbb{E}[f(t)]}{\mathbb{E}[g(t)]} \leq \rho$. Then rearranging and using linearity of expectation gives that $\mathbb{E}[f(t) - \rho g(t)] \leq 0$. Hence there must be some outcome t^* so that indeed $f(t^*) - \rho g(t^*) \leq 0$. Again rearranging gives that $\frac{f(t^*)}{g(t^*)} \leq \rho$. But it may not be true that $\mathbb{E}_{t \sim \mathcal{D}}[\frac{f(t)}{g(t)}] \leq \rho$.

decomposition $M = \sum_{i=1}^n \lambda_i u_i u_i^T$ where u_1, \dots, u_n is an orthonormal basis. Then $x = \sum_{i=1}^n \langle u_i, x \rangle u_i$ where $\langle u_i, x \rangle$ is the coordinate of x in that orthonormal basis. Then after one multiplication with M we have

$$Mx = \sum_{i=1}^n \lambda_i \langle u_i, x \rangle u_i$$

What this means is that the i th coordinate is amplified by a factor of λ_i . So repeating this procedure will make the coordinates of the (approximately) largest Eigenvalue dominate. The randomization of the initial x is only needed so that we do not accidentally start with an x that is mostly orthogonal to the Eigenvector of the largest Eigenvalue. We will prove the following:

Theorem 5.9. *Let M be a PSD matrix with Eigenvalues $\lambda_1 \geq \dots \geq \lambda_n \geq 0$ and let $0 < \varepsilon \leq \frac{1}{2}$. With constant probability one has that after $k := O(\frac{\ln(n)}{\varepsilon})$ many iterations, the returned vector y satisfies*

$$\frac{y^T M y}{\|y\|_2^2} \geq (1 - \varepsilon) \lambda_1$$

Proof. For $\ell \in \{0, \dots, k\}$, let $x^{(\ell)}$ as the vector after the ℓ -th iteration. It will be convenient to write $x = x^{(0)}$ which is the random Gaussian and $y = x^{(k)}$ which is the returned vector. Let u_i be the Eigenvector of M for the i th Eigenvalue. By induction one can easily verify that

$$y = \underbrace{MM \dots M}_{k \text{ times}} x = M^k x.$$

Note that the Eigen decomposition of M^k is simply $M^k = \sum_{i=1}^n \lambda_i^k u_i u_i^T$. We say that the starting vector x was *good* if (i) $|\langle u_1, x \rangle| \geq \frac{1}{2}$ and (ii) $\|x\|_2^2 \leq 4n$.

Claim I. *The starting point x is good with constant probability.*

Proof of Claim I. We have $\Pr[|\langle u_1, x \rangle| \geq \frac{1}{2}] = 1 - \frac{1}{\sqrt{2\pi}} \int_{-1/2}^{1/2} e^{-t^2/2} dt \geq 0.6$ and $\mathbb{E}[\|x\|_2^2] = n$ which means $\Pr[\|x\|_2^2 \geq 4n] \leq \frac{1}{4}$ by Markov's inequality⁵. The claim follows by the union bound as $1 - 0.4 - \frac{1}{4} = 0.35$. \square

Now we do the main analysis where we may assume that x is good whenever needed. First, note that

$$y^T M y = (M^k x)^T M (M^k x) = x^T M^{2k+1} x \quad \text{and} \quad y^T y = (M^k x)^T (M^k x) = x^T M^{2k} x$$

Hence we have

$$\frac{y^T M y}{\|y\|_2^2} = \frac{x^T M^{2k+1} x}{x^T M^{2k} x} \tag{5.1}$$

⁵In fact one can prove much stronger concentration such as $\Pr[\|x\|_2^2 \geq 4n] \leq e^{-\Theta(n)}$.

We will need to lower bound the nominator $x^T M^{2k+1} x$ and upper bound the denominator $x^T M^{2k} x$. It is possible that there are several Eigenvalues very close to λ_1 in which case we could not expect that y is a scalar of u_1 , but y would be close to the span of the Eigenvectors of those Eigenvalues close to λ_1 . To analyze this, let j be the index so that $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_j \geq (1 - \frac{\varepsilon}{2})\lambda_1 > \lambda_{j+1} \geq \dots \geq \lambda_n \geq 0$. In other words, the first j Eigenvalues are very close to λ_1 and the remaining Eigenvalues are far enough below λ_1 .

We can lower bound

$$x^T M^{2k+1} x = \sum_{i=1}^n \lambda_i^{2k+1} \langle u_i, x \rangle^2 \geq \lambda_1 (1 - \frac{\varepsilon}{2}) \sum_{i=1}^j \lambda_i^{2k} \langle u_i, x \rangle^2$$

by only taking the contribution from large Eigenvalues. Then we upper bound

$$\begin{aligned} x^T M^{2k} x &= \sum_{i=1}^j \lambda_i^{2k} \langle u_i, x \rangle^2 + \underbrace{\sum_{i=j+1}^n \langle u_i, x \rangle^2}_{\leq \|x\|_2^2 \leq 4n} \cdot \underbrace{\lambda_i^{2k}}_{\leq (1 - \frac{\varepsilon}{2})^{2k} \lambda_1^{2k}} \\ &\stackrel{(*)}{\leq} \sum_{i=1}^j \lambda_i^{2k} \langle u_i, x \rangle^2 + \underbrace{4n \cdot (1 - \frac{\varepsilon}{2})^{2k} \lambda_1^{2k}}_{\leq (\frac{1}{2})^2 \frac{\varepsilon}{2}} \\ &\stackrel{(**)}{\leq} (1 + \frac{\varepsilon}{2}) \sum_{i=1}^j \lambda_i^{2k} \langle u_i, x \rangle^2 \end{aligned}$$

where we use in (*) that $\|x\|_2^2 \leq 4n$ and in (**) that $|\langle u_1, x \rangle| \geq \frac{1}{2}$; both hold if we assume that x is good. In (**) we also use that $(1 - \frac{\varepsilon}{2})^k \leq \exp(-k\frac{\varepsilon}{2}) \leq \frac{\varepsilon}{32n}$ if we choose $k = \Theta(\frac{\ln(n)}{\varepsilon})$ with a suitable constant. Then we can lower bound the ratio from (5.1) as

$$\frac{x^T M^{2k+1} x}{x^T M^{2k} x} \geq \frac{\lambda_1 (1 - \frac{\varepsilon}{2}) \sum_{i=1}^j \lambda_i^{2k} \langle u_i, x \rangle^2}{(1 + \frac{\varepsilon}{2}) \sum_{i=1}^j \lambda_i^{2k} \langle u_i, x \rangle^2} \geq \lambda_1 (1 - \varepsilon)$$

We note that indeed $\frac{1 - \frac{\varepsilon}{2}}{1 + \frac{\varepsilon}{2}} \geq 1 - \varepsilon$ for all $\varepsilon \geq 0$. □

We conclude this chapter with a few comments on the power method:

- For numerical stability one typically replaces the update $x' := Mx$ by $x' := \frac{Mx}{\|Mx\|_2}$ (otherwise the length of the vector might grow/shrink enourmously).

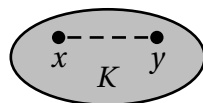
- For a non-symmetric matrix $A \in \mathbb{R}^{m \times n}$ one can find approximate left- or right-singular vectors corresponding to the largest singular value by applying the power method to AA^T or $A^T A$ (which are positive semidefinite matrices).
- The problem that we were interested in in Section 5.2 was to compute the *second smallest* Eigenvalue of the normalized Laplacian \tilde{L}_G , rather than the *largest one*. But this can be done with the following trick. Recall that $0 \leq \tilde{L}_G \leq 2I_n$. Hence $2I_n - \tilde{L}_G$ is a positive semidefinite matrix and the *second largest* Eigenvector is the one that is relevant. But we know that $D^{1/2}\mathbf{1}$ is the Eigenvector of $2I_n - \tilde{L}_G$ belonging to the largest Eigenvalue which is 2. Hence we can simply apply the power method to $M := 2I_n - \tilde{L}_G - 2 \frac{D^{1/2}\mathbf{1}\mathbf{1}^T D^{1/2}}{\|D^{1/2}\mathbf{1}\|_2^2}$.
- On first thought one might be tempted to believe that the power method also works in case that M is symmetric but not PSD. But that is not quite true. Suppose $\lambda_1, \dots, \lambda_n$ are the Eigenvalues, sorted so that $|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_n| \geq 0$. If $|\lambda_1| \geq (1 - \varepsilon)|\lambda_2|$, then indeed the method will converge with the same rate against the Eigenvector u_1 , regardless of the sign of λ_1 . But if say $|\lambda_1| = |\lambda_2| \geq (1 - \varepsilon)|\lambda_3|$ where λ_1 and λ_2 have different signs, then the method will not converge (the produced vector will lie approximately in $\text{span}\{u_1, u_2\}$ but not approach $\pm u_1$ or $\pm u_2$).

Chapter 6

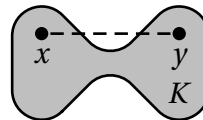
Linear programming

6.1 Convexity, polyhedra and linear programs

A set $K \subseteq \mathbb{R}^n$ is *convex* if for all $x, y \in K$ and $0 \leq \lambda \leq 1$ one has $\lambda x + (1 - \lambda)y \in K$. Intuitively, for any pair of points $x, y \in K$ in a convex set, the line segment connecting them must lie inside K .



convex



not convex

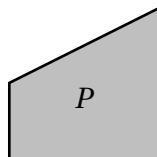
The point $\lambda x + (1 - \lambda)y$ is called a *convex combination* of x and y .

Definition 6.1. For a matrix $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, the set

$$P = \{x \in \mathbb{R}^n \mid Ax \leq b\} = \left\{ x \in \mathbb{R}^n \mid \begin{array}{l} A_1^T x \leq b_1 \\ A_2^T x \leq b_2 \\ \vdots \\ A_m^T x \leq b_m \end{array} \right\}$$

is called a *polyhedron*.

That means P is the solution to a system of finitely many linear inequalities.

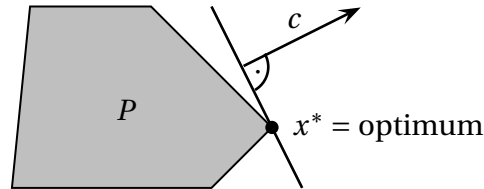


polyhedron $P \subseteq \mathbb{R}^2$

Note that every polyhedron is convex. If P is *bounded* (i.e. for some $r > 0$, $P \subseteq B_2^n(\mathbf{0}, r)$), then P is called a *polytope*. The optimization problem

$$\max \{c^T x \mid Ax \leq b\}$$

is called a *linear program (LP)*. In other words, a linear program is the problem of optimizing a linear function over a polyhedron.



LPs are fundamental problems in theoretical computer science and optimization. For example flow problems are LPs and LPs are the building block for most approximation algorithms. Linear programs have a rich structure including *duality theory* which we discuss later. For now we will focus on the question how to solve LPs algorithmically.

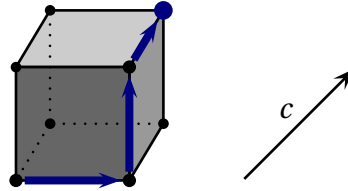
6.2 An overview over algorithms for linear programs

There are 3 main classes of algorithms to solve linear programs. We give a brief overview and then discuss one method in detail.

- **The simplex method.** Pioneered by Dantzig in 1947 (see [Dan90] for an historic account), the idea of the simplex method is to iteratively move from a *vertex*¹ $x \in P$ to another vertex $x' \in P$ with better (or equal) objective function, i.e. $c^T x' \geq c^T x$. Often there is some flexibility how to pick the next vertex and the selection rule is called the *pivot rule*. The simplex method performs excellently in practice, but to this day it is unclear how to make it provably work in polynomial time in the worst case. For most pivot rules one can construct some pathological instance (often a deformed cube) that makes the simplex method visit exponentially many vertices. Kalai [Kal92] proved that a *random* pivot rule works in subexponential time of $m^{O(\sqrt{n})}$. Spielman and Teng [ST04] proved that for certain *random* instances, the simplex method indeed takes polynomial time which

¹Formally a vertex x of P is a point in P that is not a convex combination of two distinct points that are also in P . Note that not every non-empty polyhedron $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ has vertices (the condition needed is that $\text{rank}(A) = n$). But for example each non-empty polyhedron of the form $P = \{x \in \mathbb{R}^n \mid Ax = b, x \geq \mathbf{0}\}$ does have vertices.

might be the best explanation for the observed excellent practical performance.

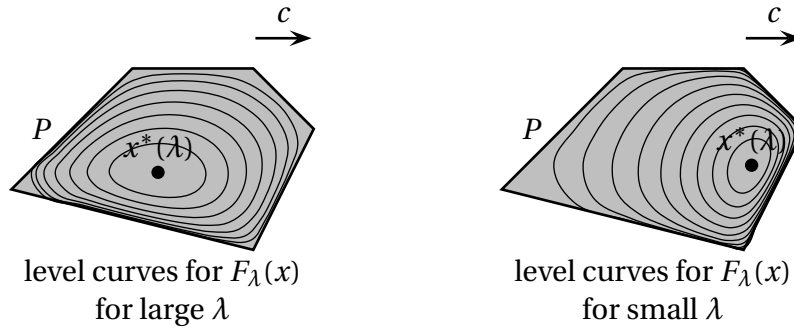


- The Ellipsoid method.** The *Ellipsoid method* was developed by Shor, Yudin, Nemirovski (1970) for nonlinear optimization; later Khachyian (1979; [Kha79]) turned it into the first polynomial time algorithm for solving linear programs. Actually the method does not optimize a linear function, it merely finds a feasible point in a convex set K . It does so by constructing a sequence $E_0 \supseteq E_1 \supseteq \dots \supseteq E_T$ of smaller and smaller ellipsoids that all contain K until a point in K is found. Then in order to actually solve the optimization question $\max\{c^T x \mid x \in K\}$, we can define $K_\beta := \{x \in \mathbb{R}^n \mid x \in K \text{ and } c^T x \geq \beta\}$ for $\beta \in \mathbb{R}$. Then using *binary search* one finds the maximal β so that $K_\beta \neq \emptyset$. The Ellipsoid method never became relevant in practice but it arguably is the cleanest and most general algorithm for LPs and convex programs. Hence we will discuss it further down in detail.
- Interior point methods.** Interior point methods (IPMs) are a family of algorithms, first developed by Karmarkar (1984; [Kar84]). IPMs also provide a polynomial time algorithm for solving linear programs, but they are competitive in practice with the simplex method (better on some classes of problems, worse on others).

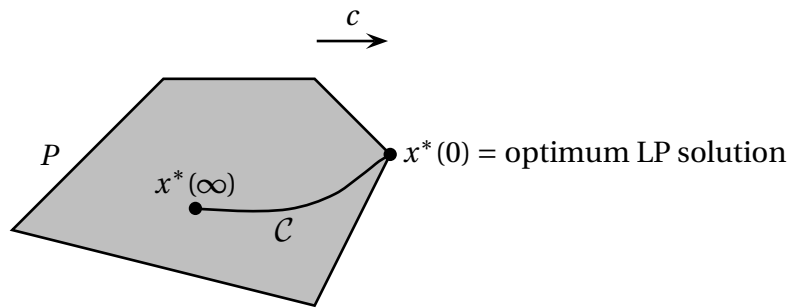
We consider a polyhedron $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ and consider the problem of maximizing $c^T x$ over P . For a parameter $\lambda > 0$, consider the *log barrier function*

$$F_\lambda(x) = c^T x + \lambda \sum_{i=1}^m \ln(b_i - a_i^T x)$$

which is defined on the interior of P and is concave. Note that the function approaches $-\infty$ if x approaches the boundary of P . Let $x^*(\lambda)$ be the unique point in P maximizing $F_\lambda(x)$. Then the smaller λ is, the smaller is the contribution of the barrier term and the optimum will approach the optimum of the LP.



The *central path* is the curve $\mathcal{C} := \{x^*(\lambda) \mid \lambda > 0\}$.



The IPM then follows the path from the *analytic center* $x^*(\infty)$ towards $x^*(0)$ in small steps (adjusting λ by a small factor in each step).

6.3 The ellipsoid method

In this section, we give some details on the Ellipsoid method, following the exposition of Korte and Vygen [KV12].

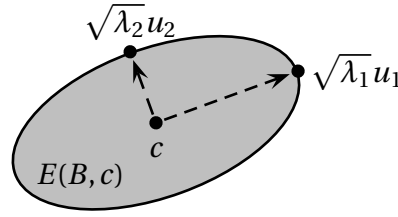
For a symmetric matrix $B \in \mathbb{R}^{n \times n}$ we write $B > 0$ (i.e. B is *positive definite*) if all Eigenvalues are strictly positive. There are several equivalent ways to define an ellipsoid. For this section it will be convenient to set

$$E(B, c) := \{x \in \mathbb{R}^n \mid (x - c)^T B^{-1} (x - c) \leq 1\}$$

where $B \in \mathbb{R}^{n \times n}$ is a matrix with $B > 0$ and $c \in \mathbb{R}^n$ is the *center* of the ellipsoid. In this notation one has $B_2^n(c, r) := E(r^2 I_n, c)$. Consider the *Eigen decomposition* of B in the form $B = \sum_{i=1}^n \lambda_i u_i u_i^T$ where $\lambda_1, \dots, \lambda_n > 0$ are the Eigenvalues and u_1, \dots, u_n are the orthonormal Eigenvectors. Then the ellipsoid can be alternatively written as

$$E(B, c) = \left\{ x \in \mathbb{R}^n \mid \sum_{i=1}^n \frac{\langle x - c, u_i \rangle^2}{\lambda_i} \leq 1 \right\}$$

In particular u_i is the *i-th axis* that has a length of $\sqrt{\lambda_i}$.



We can also see that the volume of this ellipsoid is

$$\frac{\text{Vol}_n(E(B, c))}{\text{Vol}_n(B_2^n)} = \prod_{i=1}^n \sqrt{\lambda_i} = \det(B^{1/2}) = \sqrt{\det(B)}$$

The idea behind the ellipsoid method is quite simple: say we have a target polytope $P \subseteq \mathbb{R}^n$ and we know that it is contained in some large ellipsoid $E_0 = E(A_0, c_0)$. Then we check whether $c_0 \in P$ — if yes, we are done. Otherwise, by taking an inequality $a^T x \leq b$ of P that is violated by c_0 we learn that P is contained in a *half-ellipsoid* $E_0 \cap H$ with $H := \{x \in \mathbb{R}^n \mid a^T x \leq a^T c_0\}$. Then we can explicitly compute an ellipsoid E_1 that contains $E_0 \cap H$ but has a smaller volume than E_0 . Then we iterate.

Ellipsoid method

Input: $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$ and $R > 0$ so that $P := \{x \in \mathbb{R}^n \mid Ax \leq b\}$ is contained in $B_2^n(\mathbf{0}, R)$.

Output: Point $x \in P$.

- (1) Set $B_0 := R^2 I_n$, $c_0 := \mathbf{0}$ so that $E_0 := E(B_0, c_0) = B_2^n(\mathbf{0}, R)$.
- (2) FOR $k = 0$ TO ∞

- (3) If $c_k \in P$ THEN return c_k

- (4) Select an inequality $a_k^T x \leq \beta_k$ from system $Ax \leq b$ with $a_k^T c_k > \beta_k$

- (5) Set

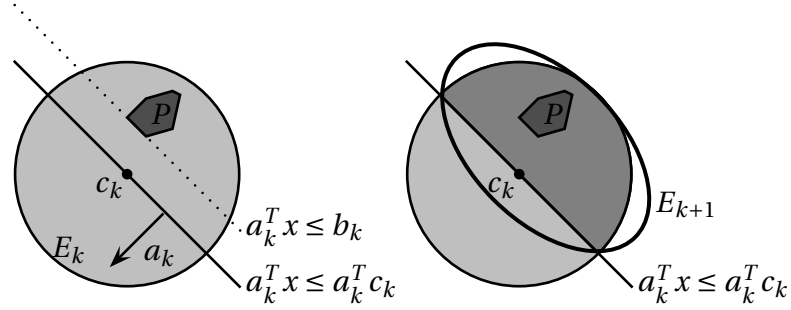
$$d_k := \frac{1}{\sqrt{a_k^T B_k a_k}} B_k a_k$$

and update

$$c_{k+1} := c_k - \frac{1}{n+1} d_k, \quad B_{k+1} := \frac{n^2}{n^2-1} \left(B_k - \frac{2}{n+1} d_k d_k^T \right)$$

and $E_{k+1} := E(B_{k+1}, c_{k+1})$.

We note that the new ellipsoid E_{k+1} is obtained by moving the center of E_k into direction $-d_k$, squeezing E_k in direction d_k and slightly expanding it in directions orthogonal to d_k . The figure for the case that E_k is a ball is as follows:



We will prove correctness of the ellipsoid method in form of the following theorem:

Theorem 6.2. *Suppose $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ is contained in $B_2^n(\mathbf{0}, R)$ and P contains some ball of radius $r > 0$. Then the ellipsoid method finds a point in P in at most $O(n^2 \log \frac{R}{r})$ many iterations.*

It suffices to prove the following two properties:

- (i) For all $k \geq 0$ one has $E_{k+1} \supseteq E_k \cap \{x \in \mathbb{R}^n \mid a_k^T x \leq a_k^T c_k\}$
- (ii) For all $k \geq 0$ one has $\frac{\text{Vol}_n(E_{k+1})}{\text{Vol}_n(E_k)} \leq \exp(-\frac{1}{2n+2})$.

By (i) we have $P \subseteq E_k$ for all k . Moreover, as P contains a radius r -ball, we know that in each iteration k we have

$$\begin{aligned}
 r^n \text{Vol}_n(B_2^n) &= \text{Vol}_n(B_2^n(\mathbf{0}, r)) \\
 &\leq \text{Vol}_n(E_k) \\
 &\stackrel{(ii)}{\leq} \exp\left(-\frac{k}{2n+2}\right) \cdot \text{Vol}_n(E_0) \\
 &= \exp\left(-\frac{k}{2n+2}\right) \cdot R^n \text{Vol}_n(B_2^n)
 \end{aligned}$$

Rearranging gives

$$k \leq n \cdot (2n+2) \ln\left(\frac{R}{r}\right)$$

which establishes the upper bound on the number of iterations in Theorem 6.2.

It remains to prove (i) and (ii). We fix some iteration k . Note that applying an affine linear transformation does not change the properties (i) and (ii). If we transform the space so that $E_k = B_2^n$ (i.e. $B_k = I_n$ and $c_k = \mathbf{0}$) and $\|a_k\|_2 = 1$ then the update formula simplifies to

$$c_{k+1} = -\frac{1}{n+1} a_k \quad \text{and} \quad B_{k+1} = \frac{n^2}{n^2-1} \left(I_n - \frac{2}{n+1} a_k a_k^T \right)$$

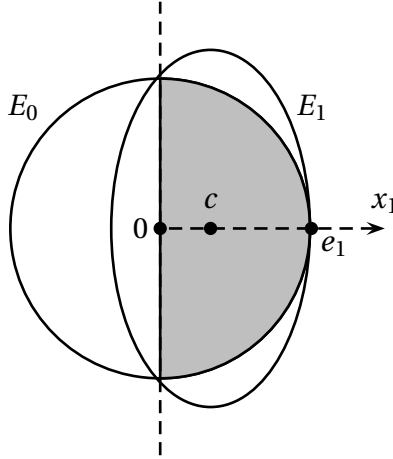
This means that in direction a_k we shrink the axis length to $\sqrt{\frac{n^2}{n^2-1}(1-\frac{2}{n+1})} = 1 - \frac{1}{n} \pm O(\frac{1}{n^2})$ while in directions orthogonal to a_k we expand the axis length to $\sqrt{\frac{n^2}{n^2-1}} = 1 + \frac{1}{2n^2} \pm O(\frac{1}{n^4})$. Note that one can prove that the update formula for the ellipsoid are chosen so that E_{k+1} is precisely the ellipsoid satisfying (i) that minimizes the volume. That ellipsoid is also called the *Löwner-John ellipsoid* with respect to the convex body $E_k \cap \{x \in \mathbb{R}^n \mid a_k^T x \leq a_k^T c_k\}$. However, in order to have a cleaner proof we will instead analyze a different ellipsoid that is slightly larger along the a_k direction so that proving (i) is easier. Without loss of generality we may also assume that $a_k = -e_1$, hence it suffices to prove the following standalone lemma:

Lemma 6.3. For $n \geq 2$, define $B \in \mathbb{R}^{n \times n}$

$$B = \text{diag}\left(\left(\frac{n}{n+1}\right)^2, \underbrace{\frac{n^2}{n^2-1}, \dots, \frac{n^2}{n^2-1}}_{n-1 \text{ entries}}\right) \quad \text{and} \quad c := \frac{1}{n+1}e_1$$

Then $E_0 := B_2^n$ and $E_1 := E(B, c)$ satisfy

- (i) One has $E_1 \supseteq E_0 \cap \{x \in \mathbb{R}^n \mid x_1 \geq 0\}$
- (ii) One has $\frac{\text{Vol}_n(E_1)}{\text{Vol}_n(E_0)} \leq \exp(-\frac{1}{2n+2})$.



Proof. First note that the ellipsoid E_1 is of the form

$$\begin{aligned} E_1 &= \{x \in \mathbb{R}^n \mid (x - c)^T B^{-1} (x - c) \leq 1\} \\ &= \left\{ x \in \mathbb{R}^n \mid \left(\frac{n+1}{n}\right)^2 \left(x_1 - \frac{1}{n+1}\right)^2 + \frac{n^2-1}{n^2} \sum_{i=2}^n x_i^2 \leq 1 \right\} \end{aligned}$$

To show (i), we prove that for $x \in \mathbb{R}^n$ with $\|x\|_2 \leq 1$ and $x_1 \geq 0$ one has $x \in E_1$. And indeed

$$\begin{aligned} & \left(\frac{n+1}{n}\right)^2 \left(x_1 - \frac{1}{n+1}\right)^2 + \frac{n^2-1}{n^2} \underbrace{\sum_{i=2}^n x_i^2}_{\leq 1-x_1^2} \\ & \leq \left(\left(\frac{n+1}{n}\right)^2 - \frac{n^2-1}{n^2}\right)x_1^2 - 2\frac{1}{n+1}\left(\frac{n+1}{n}\right)^2 x_1 + \left(\left(\frac{n+1}{n}\right)^2 \frac{1}{(n+1)^2} + \frac{n^2-1}{n^2}\right) \\ & = \frac{2n+2}{n^2} \cdot \underbrace{(x_1^2 - x_1)}_{\leq 0} + 1 \leq 1 \end{aligned}$$

using that $0 \leq x_1 \leq 1$. To verify claim (ii) we estimate that

$$\begin{aligned} \frac{\text{Vol}_n(E_1)}{\text{Vol}_n(E_0)} &= \sqrt{\det(B)} = \prod_{i=1}^n \sqrt{B_{ii}} \\ &= \frac{n}{n+1} \cdot \left(\frac{n^2}{n^2-1}\right)^{(n-1)/2} = \left(1 - \frac{1}{n+1}\right) \cdot \left(1 + \frac{1}{n^2-1}\right)^{(n-1)/2} \\ &\leq \exp\left(-\frac{1}{n+1}\right) \cdot \exp\left(\frac{1}{n^2-1} \cdot \frac{n-1}{2}\right) = \exp\left(-\frac{1}{2(n+1)}\right) \end{aligned}$$

Here we use that B is diagonal and the fact that $1 + y \leq e^y$ for all $y \in \mathbb{R}$. \square

There are some further technical issues that need to be resolved in order to make the Ellipsoid method work:

- (a) Make P bounded and find an enclosing ball.
- (b) Make P full-dimensional and find a lower bound on a ball inside P .
- (c) The algorithm contains square roots; round them to rational numbers and incorporate the error.

We refer again to the book of Korte and Vygen [KV12] for details. We conclude:

Theorem 6.4. *Let $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$, $c \in \mathbb{Q}^n$. Then the LP $\max\{c^T x \mid Ax \leq b\}$ can be solved in time polynomial in the bit encoding length of A, b, c .*

6.3.1 The ellipsoid method with a separation oracle

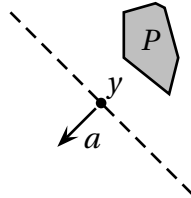
Consider again a polytope $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ with $A \in \mathbb{Q}^{m \times n}$ and $b \in \mathbb{Q}^m$ with $B_2^n(c^*, r) \subseteq P \subseteq B_2^n(\mathbf{0}, R)$ for some unknown point $c^* \in \mathbb{R}^n$ and $r \leq R$. In Theorem 6.2 we have seen that the ellipsoid method takes $O(n^2 \log \frac{R}{r})$ many iterations

to find a feasible point. That means the number of iterations does not depend on the number m of inequalities. Instead of maintaining an explicit list of all m inequalities in the system $Ax \leq b$, it would suffice if in every iteration, the algorithm could determine *one* inequality violated by the current center c_k . This is called the *separation problem* for P .

SEPARATION PROBLEM FOR POLYHEDRON $P \subseteq \mathbb{R}^n$

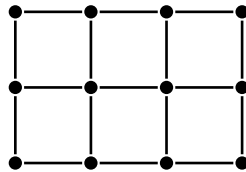
Input: Point $y \in \mathbb{Q}^n$

Goal: Find a $a \in \mathbb{Q}^n$ with $a^T y > a^T x \forall x \in P$ or assert $y \in P$.

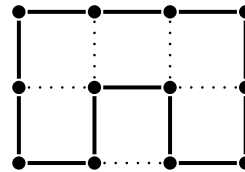


Hence, if for a polytope P (or more generally a convex set) we can solve the separation problem in polynomial time, then we can also find a point in it. We want to demonstrate this using an example.

The *Travelling salesperson problem (TSP)* is the following: given an undirected graph $G = (V, E)$ and a cost function $c : E \rightarrow \mathbb{R}_{\geq 0}$, find the minimum cost *tour* / *Hamiltonian circuit* $F \subseteq E$. Here a tour is an edge set that is a connected cycle visiting all nodes exactly once.



graph G

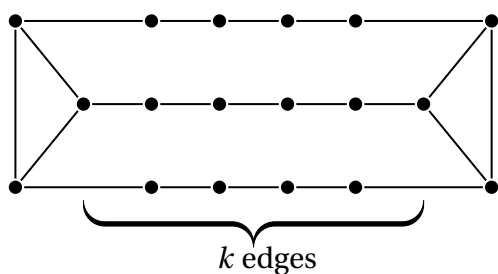
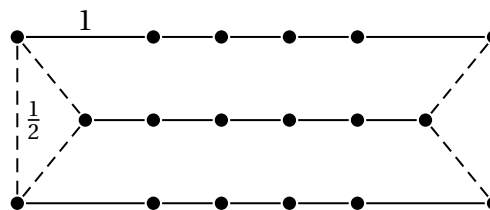


tour

The problem is **NP-hard**. But the problem has a very useful linear programming *relaxation* which is called the *subtour elimination LP*.

$$\begin{aligned} \min \sum_{e \in E} c_e x_e & \quad (LP) \\ x(\delta(v)) &= 2 \quad \forall v \in V \\ x(\delta(S)) &\geq 2 \quad \forall \emptyset \subset S \subset V \\ x_e &\geq 0 \quad \forall e \in E \end{aligned}$$

For every tour F , the characteristic vector $\mathbf{1}_F$ is a feasible solution to the LP. But in reverse the LP might have solutions that are cheaper than any actual tour even in a complete graph where c is a metric. Here is a well known construction that gives a gap of $4/3$.

graph H 

fractional solution x of cost $(3 \pm o(1))k$
 best integral solution has cost $(4 \pm o(1))k$
 in (G, c) where $c(i, j) :=$ shortest path in H
 between i and j .

But in that important case the LP is not too far off:

Theorem 6.5 (Karlin, Klein, Oveis Gharan [KKG21, KKG22]). *If G is a complete graph and c is a metric (i.e. the cost satisfy the triangle inequality), then there is a tour of value at most $(\frac{3}{2} - 10^{-36})$ times the value of the LP.*

This work² improved a longstanding bound of $3/2$ due to Christofides [Chr76]. Note that there are roughly $2^{|V|}$ many constraints in the LP so it is not obvious in the first place that one could even solve the LP efficiently. That is the aspect that we want to explain in more detail.

Theorem 6.6. *The subtour elimination LP can be solved in polynomial time.*

Proof. Let P_β be the set of solutions $x \in \mathbb{R}^E$ to (LP) so that $c^T x \leq \beta$. By the Ellipsoid method it suffices to show that the separation problem for P_β can be solved efficiently. Let $x^* \in \mathbb{R}^E$ be a point for which we need to decide whether $x^* \in P_\beta$ and if not, we need to produce a violated inequality. First we check whether $c^T x^* \leq \beta$, $x^*(\delta(v)) = 2$ and $x_e^* \geq 0$ by going explicitly through all such inequalities. Suppose these are all satisfied. Then the remaining problem is: Given $x^* \in \mathbb{R}_{\geq 0}^E$, find the set $\emptyset \subset S \subset V$ that minimizes $x^*(\delta(S))$. This is exactly the MinCut problem which we know from Chapter 1 can be solved in polynomial time! \square

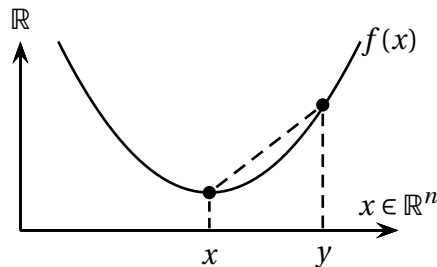
²See also the nicely written Quanta article under <https://www.quantamagazine.org/computer-scientists-break-traveling-salesperson-record-20201008/>.

6.4 Convex programming

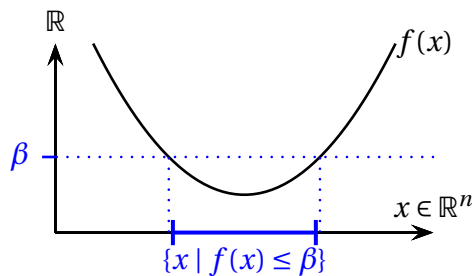
A function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is *convex* if

$$f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y) \quad \forall x, y \in \mathbb{R}^n \quad \forall 0 \leq \lambda \leq 1.$$

In other words, for every x and y , the line segment between $(x, f(x))$ and $(y, f(y))$ lies above the graph of f .



If f is convex, then for all $\beta \in \mathbb{R}$, the *level set* $\{x \in \mathbb{R}^n \mid f(x) \leq \beta\}$ is convex.



For example every linear function $f(x) = \langle a, x \rangle$ (with $a \in \mathbb{R}^n$) is convex. Here is another example:

Lemma 6.7. Let $\|\cdot\| : \mathbb{R}^n \rightarrow \mathbb{R}_{\geq 0}$ be a norm. Then $\|\cdot\|$ is convex.

Proof. Let $x, y \in \mathbb{R}^n$ and $0 \leq \lambda \leq 1$. Then we verify that

$$\|\lambda x + (1-\lambda)y\| \stackrel{\text{triangle ineq.}}{\leq} \|\lambda x\| + \|(1-\lambda)y\| \stackrel{\text{homog.}}{=} \lambda\|x\| + (1-\lambda)\|y\|$$

□

The reader may recall the following fact from the 1-dimensional setting: if $f : \mathbb{R} \rightarrow \mathbb{R}$ is twice differentiable then f is convex if and only if $f''(x) \geq 0$ for all $x \in \mathbb{R}$. This fact generalizes to the multivariate case. For $f : \mathbb{R}^n \rightarrow \mathbb{R}$ we write

$\nabla f(x) \in \mathbb{R}^n$ as the *gradient* and $\nabla^2 f(x) \in \mathbb{R}^{n \times n}$ is the *Hessian*³. Note that if all 2nd partial derivatives are continuous then $\nabla^2 f(x)$ is symmetric.

Theorem 6.8. *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a function with continuous 2nd derivatives everywhere. Then f is convex if and only if*

$$\nabla^2 f(x) \geq 0 \quad \forall x \in \mathbb{R}^n$$

We leave the proof as an exercise.

Example 6.9. Let $A \in \mathbb{R}^{n \times n}$ be a symmetric matrix and let $f(x) := x^T A x$. Then $\nabla f(x) = 2Ax$ and $\nabla^2 f(x) = 2A$. Hence the function f is convex if and only if $A \geq 0$.

A (general) *convex program* is of the form

$$\begin{aligned} \min f_0(x) & \quad (CP) \\ f_1(x) & \leq b_1 \\ & \vdots \\ f_m(x) & \leq b_m \end{aligned}$$

where $f_0, f_1, \dots, f_m : \mathbb{R}^n \rightarrow \mathbb{R}$ are convex functions.

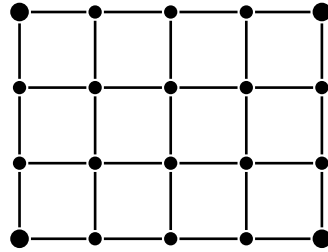
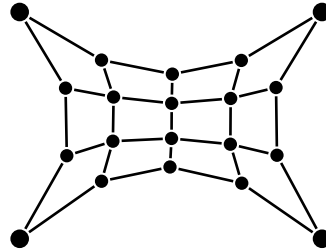
Example 6.10. Here is an example convex program from graph drawing. Consider an undirected graph $G = (V, E)$ for which we want to construct a good 2-dimensional drawing based on the following physics-inspired interpretation: we select a subset $U \subseteq V$ of vertices that we “nail” to a board. Then we install springs along the edges of the graph and wait until the system reaches a *minimum energy state*. This can be modelled as a convex program. Let $b(v) \in \mathbb{R}^2$ be the fixed coordinates for $v \in U$. We use variables $x(v) \in \mathbb{R}^2$ for all $v \in V$. Then

$$\min \left\{ \sum_{\{u,v\} \in E} \|x(u) - x(v)\|_2^2 \mid x(v) = b(v) \quad \forall v \in U \right\}$$

is the convex program⁴ that provides the minimum energy state.

³Formally, the gradient is the vector of partial (first) derivatives, i.e. $\nabla f(x^*) := \left(\frac{\partial f}{\partial x_1}(x^*), \dots, \frac{\partial f}{\partial x_n}(x^*) \right)$. The Hessian (matrix) is the $n \times n$ matrix with all the second partial derivatives, i.e. $\nabla^2 f(x^*) := \left(\frac{\partial^2 f}{\partial x_i \partial x_j}(x^*) \right)_{i,j \in [n]}$.

⁴One can think of the constraints as the convex constraints $x(v)_i \leq b(v)_i$ and $-x(v)_i \leq -b(v)_i$ for $i \in \{1, 2\}$.

graph G Minimum energy state
fixing the 4 corners

The name “convex program” is justified because of the following:

Lemma 6.11. *For any $b_0 \in \mathbb{R}$, the set $K = \{x \in \mathbb{R}^n \mid f_0(x) \leq b_0, f_i(x) \leq b_i \forall i = 1, \dots, m\}$ of solutions to (CP) that have value b_0 is convex.*

Proof. We already know that the sets $S_i := \{x \in \mathbb{R}^n \mid f_i(x) \leq b_i\}$ are convex. Then $K = S_0 \cap S_1 \cap \dots \cap S_m$. Since the intersection of convex sets is convex, the claim follows. \square

In particular the set of optimum solutions is convex. Note that in (CP) if one replaces min with max or \leq with \geq or $=$ then this convexity property may be lost. We will show that convex programs can be solved in polynomial time — modulo some technicalities that we skip here⁵.

Theorem 6.13 (Informal). *One can solve any convex program (CP) in polynomial time as long as for every $x \in \mathbb{R}^n$ and $i \in \{0, \dots, m\}$ one can compute $f_i(x)$ and $\nabla f_i(x)$ efficiently.*

Proof. Using binary search on the function value it suffices to find a feasible point in

$$K := \{x \in \mathbb{R}^n \mid f_i(x) \leq b_i \forall i = 0, \dots, m\}$$

or assert that K is empty. We use the Ellipsoid method to do this, starting with an ball of large enough radius. From Section 6.3.1 we know that it suffices to solve the Separating Hyperplane Oracle for K at some given point y . By assumption

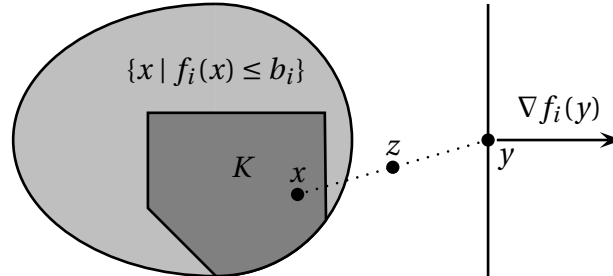
⁵The precise statement that one could prove is the following:

Theorem 6.12. *Consider a convex program (CP) so that (i) all feasible solutions are contained in $B_2^n(\mathbf{0}, R)$ and the functions f_i are R -Lipschitz continuous, (ii) the functions f_i have continuous 2nd derivatives, and (iii) for all $x \in \mathbb{R}^n$ and $i \in \{0, \dots, m\}$ we can compute $f_i(x)$ and $\nabla f_i(x)$ in polynomial time. Then in time $\text{poly}(n, m, \log(R), \log(\frac{1}{\epsilon}))$ one can compute an x^* with $f_i(x^*) \leq b_i + \epsilon$ for $i \in [m]$ and $f_0(x^*) \leq \text{opt} + \epsilon$.*

we can evaluate $f_i(y)$ at any i and hence we can decide whether $y \in K$ or not. Suppose that $y \notin K$ and let i be a constraint with $f_i(y) > b_i$.

Claim I. $\nabla f_i(y)$ is the normal vector separating y from K .

Proof of Claim I. Fix a point $x \in K$. We need to prove that $\langle \nabla f_i(y), y \rangle > \langle \nabla f_i(y), x \rangle$.



By (a variant of) *Taylor's Theorem*, there is a point z on the line segment between x and y so that

$$f_i(x) \stackrel{\text{Taylor}}{=} f_i(y) + (x-y)^T \nabla f_i(y) + \underbrace{(x-y)^T \nabla^2 f_i(z) (x-y)}_{\geq 0} \geq f_i(y) + \langle \nabla f_i(y), x-y \rangle$$

Here we use that because f_i is convex, the Hessian $\nabla^2 f_i(z)$ is PSD. Rearranging gives

$$\langle \nabla f_i(y), y \rangle \geq \langle \nabla f_i(y), x \rangle + \underbrace{f_i(y)}_{> b_i} - \underbrace{f_i(x)}_{\leq b_i} > \langle \nabla f_i(y), x \rangle \quad \square$$

As the gradient $\nabla f_i(y)$ can be computed efficiently, the claim follows. \square

To understand why there may be technical issues and in particular some error term $\varepsilon > 0$ may be needed, consider for example the 1-dimensional convex program $\min\{-x : x \in \mathbb{R} \text{ and } x^2 \leq 2\}$. Then the optimum solution is $\sqrt{2}$ (i.e. irrational) even though the input data is rational.

6.5 Semidefinite programming

A *semidefinite program* is of the form:

$$\begin{aligned} \min \langle C, X \rangle \\ \langle A_k, X \rangle &\leq b_k \quad \forall k = 1, \dots, m \\ X &\geq 0 \end{aligned}$$

where $C, A_1, \dots, A_m \in \mathbb{R}^{n \times n}$. In other words, we do not optimize over vectors but over $n \times n$ PSD matrices. Note that by Lemma 4.5.(b), the set of PSD matrices is of the form

$$\begin{aligned} \mathcal{C} &:= \{X \in \mathbb{R}^{n \times n} \mid X \geq 0\} \\ &= \{X \in \mathbb{R}^{n \times n} \mid X \text{ symmetric and } \langle X, uu^T \rangle \geq 0 \ \forall u \in \mathbb{R}^n\} \end{aligned}$$

and hence it is convex. In particular the separation problem for a matrix $X^* \notin \mathcal{C}$ can be solved by computing the Eigenvector v for the smallest Eigenvalue of X^* . Then $\langle X^*, vv^T \rangle < 0$ while $\langle X, vv^T \rangle \geq 0$ for all $X \in \mathcal{C}$. Consequently semidefinite programs can be solved in polynomial time (again modulo technicalities and up to some numerical error).

Recall that $Y \geq 0$ is equivalent to $Y_{ij} = \langle v_i, v_j \rangle$ for some vectors v_i . Making that substitution in an SDP results in an equivalent program that is called *vector program*.

SDP:

$$\begin{aligned} \max \sum_{i,j} C_{ij} Y_{ij} \\ \sum_{i,j} A_{ij}^k \cdot Y_{ij} &\leq b_k \quad \forall k \\ Y &\text{ sym.} \\ Y &\geq 0 \end{aligned}$$

Vector program:

$$\begin{aligned} \max \sum_{i,j} C_{ij} \langle v_i, v_j \rangle \\ \sum_{i,j} A_{ij}^k \cdot \langle v_i, v_j \rangle &\leq b_k \quad \forall k \\ v_i &\in \mathbb{R}^r \quad \forall i \end{aligned}$$

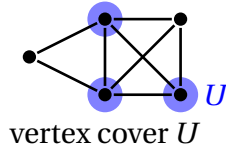
Curiously, the right hand side vector program is *not* convex, but rather is equivalent to a convex program (the SDP) and hence can be solved in polynomial time. Also note that one cannot choose the dimension r of the vectors. In fact, solving a vector program subject to the rank restriction $r = 1$ is again **NP**-hard.

6.6 Rounding linear programs and SDPs

We want to demonstrate the usefulness of linear programs and convex programs to solve inherently discrete problems approximately.

6.6.1 Vertex Cover

Consider an undirected graph $G = (V, E)$ with vertex weights $w : V \rightarrow \mathbb{R}_{\geq 0}$. A set $U \subseteq V$ is a *vertex cover* if $e \cap U \neq \emptyset$ for all $e \in E$.



The *minimum vertex cover problem* is the problem of finding the vertex cover U that minimizes the weight $w(U) := \sum_{i \in U} w(i)$. We can prove the following:

Theorem 6.14. *One can find a vertex cover U of weight $w(U) \leq 2OPT$ in polynomial time where OPT is the value of the optimum solution.*

Proof. We want to write down an *integer linear program (ILP)* that captures the vertex cover problem. It seems natural to introduce variables $x_i \in \{0, 1\}$ with the interpretation

$$x_i = \begin{cases} 1 & \text{if } i \text{ is included in the vertex cover} \\ 0 & \text{otherwise} \end{cases}$$

Then we can write the vertex cover problem as

$$OPT := \min \left\{ \sum_{i \in V} w_i x_i \mid x_i + x_j \geq 1 \ \forall \{i, j\} \in E; \ x_i \in \{0, 1\} \ \forall i \in V \right\} \quad (*)$$

where it was important that the constraints and objective function were linear⁶ in x . But solving an ILP is **NP-hard** in general so we cannot actually solve (*). But we can relax the constraint $x_i \in \{0, 1\}$ to $0 \leq x_i \leq 1$ and make it a linear program:

$$LP := \min \left\{ \sum_{i \in V} w_i x_i \mid x_i + x_j \geq 1 \ \forall \{i, j\} \in E; \ 0 \leq x_i \leq 1 \ \forall i \in V \right\} \quad (**)$$

We can solve this LP and denote its optimum solution by $x^* \in [0, 1]^V$. Since the LP is a relaxation we know that $LP \leq OPT$. But on the other hand the vector x^* is (in general) fractional and we still need to extract a vertex cover. We set

$$U := \left\{ i \in V \mid x_i^* \geq \frac{1}{2} \right\}$$

as our solution. Clearly U is a vertex cover because for every edge $\{i, j\} \in E$ we have $x_i^* + x_j^* \geq 1$ and so $\max\{x_i^*, x_j^*\} \geq \frac{1}{2}$. The cost of that solution U is

$$\sum_{i \in U} w_i \leq \sum_{i \in U} \underbrace{2x_i^*}_{\geq 1} w_i = 2LP \leq 2OPT$$

□

⁶We could have gotten away with convex constraints, but linear is easier.

We want to give a few comments. The algorithm above is called a *2-approximation algorithm* for minimum vertex cover. Many approximation algorithms in the literature follow the same scheme as above:

- (1) Solve the LP and let x^* denote the optimum fractional solution.
- (2) Somehow round x^* to a feasible solution that is not much more costly.

When designing an approximation algorithm the main questions are: What LP relaxation to use (sometimes there is an obvious one; sometimes not)? How to round the fractional solution?

For a general minimization problem and a fixed LP relaxation, the *integrality gap* is the ratio

$$\sup_{\text{instances } \mathcal{I}} \frac{\text{optimum of } \mathcal{I}}{\text{LP value of } \mathcal{I}}$$

The integrality gap is usually a natural barrier for those types of algorithms. For example the argument above shows that the integrality gap for the LP (***) is at most 2. But one can prove that the integrality gap is also not smaller than 2. For example let G be the complete graph on n vertices with unit vertex weights. Then setting $x_i^* := \frac{1}{2}$ gives a fractional solution of value $\frac{n}{2}$. But any actual vertex cover must include all but one vertex. Hence the integrality gap is at least $\frac{n-1}{n/2} \rightarrow 2$. Moreover, assuming the Unique Games Conjecture of Khot [Kho02] there is no polynomial time $(2 - \varepsilon)$ -approximation for any constant $\varepsilon > 0$ [KR08]. That means, the simple approximation algorithm from above is likely optimal.

6.6.2 Set Cover

Next, we consider the (*weighted*) *Set Cover problem* where we are given a family of sets $S_1, \dots, S_m \subseteq [n]$ with weights $w_1, \dots, w_m \geq 0$. The goal is to select a minimum cost subfamily whose union is the universe $[n]$, i.e. the goal is to solve $\min\{\sum_{i \in I} w_i \mid \bigcup_{i \in I} S_i = [n]\}$. Similar to the vertex cover case we want to develop an LP-based approximation algorithm. We use decision variables

$$x_i = \begin{cases} 1 & \text{if } i \in I \\ 0 & \text{otherwise} \end{cases}$$

Then we use the LP

$$\begin{aligned} \min \sum_{i=1}^m w_i x_i & \quad (LP) \\ \sum_{i: j \in S_i} x_i & \geq 1 \quad \forall j \in [n] \\ 0 \leq x_i & \leq 1 \quad \forall i \in [m] \end{aligned}$$

Note that the constraint says that of all sets covering an element j , we must take at least one. Again we can optimally solve the LP and denote x^* as the optimum solution and we denote $LP := \sum_{i=1}^m w_i x_i^*$ as its value. The interesting part is how to extract an actual covering from x^* . We use a technique that is called *randomized rounding*. More precisely, for a parameter $\alpha > 0$ that we determine later we do the following:

- (1) *Sampling*: FOR $i = 1, \dots, m$ DO Pick S_i with probability $\min\{\alpha \cdot x_i^*, 1\}$
- (2) *Repairing*: FOR every not covered element $j \in [n]$ pick the cheapest set containing j

That means in (1) we randomly sample each set S_i proportional to its fractional value x_i^* . After that it may be that some elements are uncovered and we simply cover each (pessimistically using one set per remaining element).

Theorem 6.15. *For a suitable choice of α , the expected cost of the solution is at most $(\ln(n) + 1) \cdot LP$.*

Proof. The expected cost that we pay for sets included in (1) is $\sum_{i=1}^m \min\{\alpha x_i^*, 1\} w_i \leq \alpha \cdot LP$. Next we analyze the probability that elements are covered:

Claim I. *Fix an element $j \in U$. Then j is covered in (1) with probability at least $1 - e^{-\alpha}$.*

Proof of Claim I. If there is an index i with $j \in S_i$ and $\alpha x_i^* \geq 1$, then the probability in question is even 1. So suppose otherwise. Then

$$\begin{aligned} \Pr[j \text{ not covered in (1)}] &= \prod_{i: j \in S_i} \Pr[S_i \text{ not picked in (1)}] \\ &\leq \prod_{i: j \in S_i} (1 - \alpha \cdot x_i^*) \\ &\stackrel{1+y \leq e^y}{\leq} \prod_{i: j \in S_i} e^{-\alpha \cdot x_i^*} = \exp\left(-\alpha \cdot \underbrace{\sum_{i: j \in S_i} x_i^*}_{\geq 1}\right) \leq e^{-\alpha} \quad \square \end{aligned}$$

Then in the unlikely case that j is not covered in (1) we cover it in (2) by a set that has cost at most LP . Summing over the n elements we get that the

$$\begin{aligned} \mathbb{E}[\text{cost of solution}] &\leq \mathbb{E}[\text{cost in (1)}] + \mathbb{E}[\text{cost in (2)}] \\ &\leq \alpha LP + n \cdot e^{-\alpha} LP \stackrel{\alpha := \ln(n)}{=} (\ln(n) + 1) \cdot LP \end{aligned}$$

That concludes the argument. □

Again, the algorithm is mostly optimal. Feige [Fei98] proved that assuming $\mathbf{NP} \not\subseteq \mathbf{DTIME}(n^{O(\log \log n)})$ ⁷, there is no $(1 - \varepsilon) \ln(n)$ -approximation algorithm for set cover for any constant $\varepsilon > 0$. However if all sets have a bounded size of $|S_i| \leq k$, then the ratio from above can be improved to $\ln(k) + 1$.

6.6.3 MaxCut

Recall that for the MaxCut problem we are given an undirected graph $G = (V, E)$ with edge weights $w : E \rightarrow \mathbb{R}_{\geq 0}$ and the goal is to find a cut $S \subseteq V$ that maximizes the weights $w(\delta(S))$ of the separated edges. We have seen an additive approximation algorithm in Section 4.6. Here we will discuss the seminal algorithm of Goemans and Williamson [GW95] that provides a multiplicative approximation factor.

We consider the following semidefinite program with equivalent vector program:

<p>SDP:</p> $\begin{aligned} \max \quad & \frac{1}{2} \sum_{\{i,j\} \in E} w_{ij} \cdot (1 - X_{ij}) \\ X & \geq 0 \\ X_{ii} & = 1 \quad \forall i \in V \\ X & \in \mathbb{R}^{n \times n} \end{aligned}$	<p>Vector program:</p> $\begin{aligned} \max \quad & \frac{1}{2} \sum_{\{i,j\} \in E} w_{ij} \cdot (1 - \langle u_i, u_j \rangle) \\ \ u_i\ _2 & = 1 \quad \forall i \in V \\ u_i & \in \mathbb{R}^r \end{aligned}$
---	--

We verify that these programs indeed provide a relaxation.

Lemma 6.16. *If $S^* \subseteq V$ is optimum solution for MaxCut, then $SDP \geq w(\delta(S^*))$.*

Proof. We choose vectors in dimension $r := 1$ and define $u_i \in \mathbb{R}^1$ by

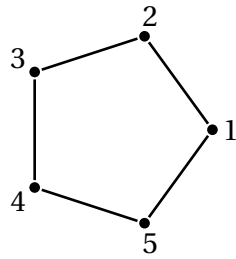
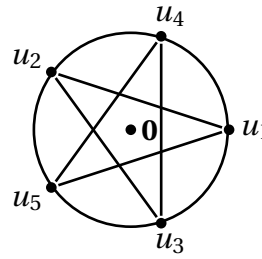
$$u_i := \begin{cases} 1 & \text{if } i \in S^* \\ -1 & \text{if } i \in V \setminus S^* \end{cases}$$

□

Of course the SDP relaxation is not exact. But it is a lot less obvious how to come up with integrality gap constructions.

Example 6.17. Consider the graph G which is a cycle on 5 nodes and unit weights $w(e) = 1$. The optimum MaxCut is 4. On the other hand, choosing $u_i \in \mathbb{R}^2$ with $u_i := (\cos(\frac{4i\pi}{5}), \sin(\frac{4i\pi}{5}))$ we get a vector program solution of value $5 \cdot \frac{1}{2}(1 - \cos(\frac{4}{5}\pi)) \approx 4.522$.

⁷This assumption is slightly stronger than $\mathbf{NP} \neq \mathbf{P}$ but widely believed.

graph G 

SDP solution:

We need an algorithm that can transform a MaxCut SDP solution into an actual cut $\delta(S)$ while being close in terms of the objective function value. The algorithm is quite simple: randomly partition \mathbb{R}^r into two halfspaces; then let S be the vertices whose vectors u_i lie in one of those halfspaces.

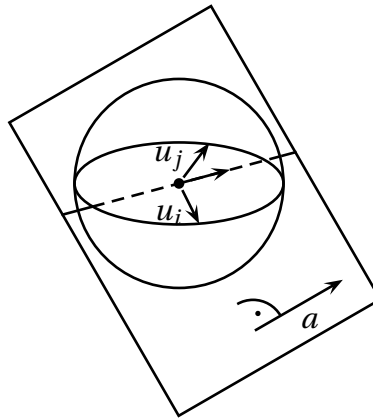
Hyperplane Rounding algorithm

Input: Graph $G = (V, E)$ and edge weights $w : E \rightarrow \mathbb{R}_{\geq 0}$

Output: Set $S \subseteq V$

- (1) Solve the MaxCut SDP
- (2) Sample a random standard Gaussian $a \in \mathbb{R}^r$
- (3) Define $S := \{i \in V \mid \langle a, u_i \rangle \geq 0\}$

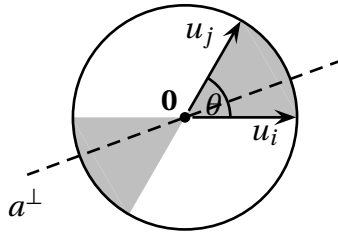
Note that the algorithm that we will analyze is *randomized* and we will prove that the *expected* value of the produced cut is good.



Lemma 6.18. For $\{i, j\} \in E$ one has $\Pr[\{i, j\} \in \delta(S)] = \frac{1}{\pi} \arccos(\langle u_i, u_j \rangle)$.

Proof. First note that the angle between vectors u_i and u_j is exactly $\theta := \arccos(\langle u_i, u_j \rangle)$ (as $\langle u_i, u_j \rangle = \cos(\theta)$). Next, note that considering only the edge $\{i, j\}$ the random experiment is equivalent to drawing a random unit vector a in the 2-dimensional⁸ space $U := \text{span}\{u_i, u_j\}$. Then we can verify that indeed $\Pr[u_i, u_j \text{ separated}] = \frac{2\theta}{2\pi}$.

⁸The case that $u_i = \pm u_j$ is obvious.

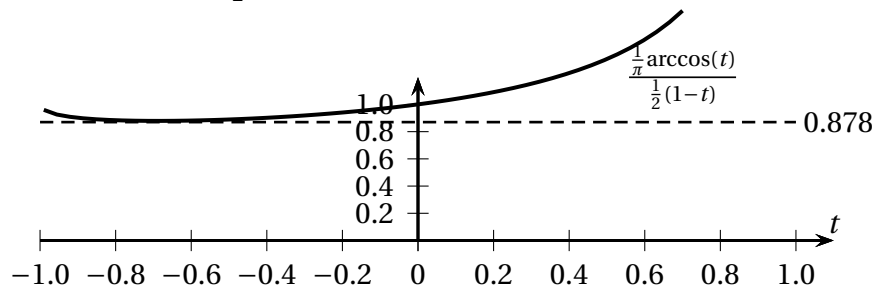


□

Theorem 6.19. One has $\mathbb{E}[w(\delta(S))] \geq 0.878 \cdot \text{SDP}$.

Proof. Fix an edge $e = \{i, j\} \in E$ and abbreviate $t := \langle u_i, u_j \rangle$. Then the contribution of that edge to the SDP objective function is $w_{ij} \cdot \frac{1}{2}(1 - t)$. On the other hand, the expected contribution of e to the cut value is $w_{ij} \cdot \Pr[\{i, j\} \in \delta(S)] = w_{ij} \cdot \frac{1}{\pi} \arccos(t)$. We can numerically verify that the ratio is indeed

$$\frac{\frac{1}{\pi} \arccos(t)}{\frac{1}{2}(1 - t)} \geq 0.878 \quad \forall t \in [-1, 1]$$



The claim then follows by *linearity of expectation*. □

Assuming the Unique Games Conjecture, there is no polynomial time algorithm that improves the ≈ 0.878 ratio for MaxCut, see [KKMO07].

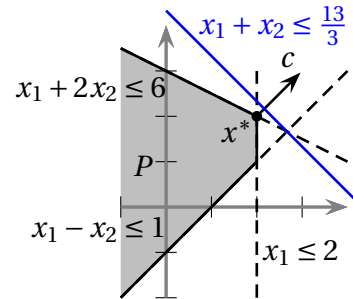
6.7 LP duality

The key structural concept in linear optimization is *duality*. We want to motivate this with an example. Consider the linear program

$$\max\{x_1 + x_2 \mid x_1 + 2x_2 \leq 6, x_1 \leq 2, x_1 - x_2 \leq 1\}$$

(which is of the form $\max\{c^T x \mid Ax \leq b\}$). First, let us make the following observation: if we add up non-negative multiples of feasible inequalities, then we again obtain an inequality that is valid for each solution x of the LP. For example we can add up the inequalities in the following way:

$$\begin{array}{rcl}
 \frac{2}{3} \cdot (& x_1 & +2x_2 \leq 6) \\
 0 \cdot (& x_1 & \leq 2) \\
 \frac{1}{3} \cdot (& x_1 & -x_2 \leq 1) \\
 \hline
 & x_1 & +x_2 \leq \frac{13}{3} \approx 4.33
 \end{array}$$



Accidentally, the feasible inequality $x_1 + x_2 \leq \frac{13}{3}$ that we obtain has the objective function as normal vector. Hence for each $(x_1, x_2) \in P$ we must have $c^T x \leq \frac{13}{3}$, which provides an *upper bound* on the value of the LP. Inspecting the picture, we quickly see that optimum solution is $x^* = (2, 2)$ with objective function $c^T x^* = 4$. Now, let's generalize our observations. Consider the following pair of linear programs

$$\begin{array}{ll}
 \text{primal } (P) : & \max\{c^T x \mid Ax \leq b\} \\
 \text{dual } (D) : & \min\{b^T y \mid A^T y = c, y \geq \mathbf{0}\}
 \end{array}$$

The dual LP is searching for inequalities $(y^T A)x \leq y^T b$ that are feasible for any primal solution x ; moreover it is looking for a feasible inequality so that the normal vector $y^T A = c^T$ (which is the same as $A^T y = c$) is the objective function, so that $y^T b$ is an upper bound on the primal LP. In other words: the dual LP is searching for the best upper bound for the primal LP.

Theorem 6.20 (Weak duality Theorem). *One has $(P) \leq (D)$.*

Proof. Fix a pair (x, y) with $Ax \leq b$, $A^T y = c$ and $y \geq \mathbf{0}$. Then one has

$$\underbrace{c^T}_{=y^T A} x = \underbrace{y^T}_{\geq \mathbf{0}} \underbrace{Ax}_{\leq b} \leq y^T b.$$

□

Interestingly, one can always find a combination of primal inequalities to certify the optimum value:

Theorem 6.21 (Strong Duality Theorem). *One has $(P) = (D)$. More precisely, one has*

$$\max\{c^T x : Ax \leq b\} = \min\{b^T y : A^T y = c, y \geq \mathbf{0}\}$$

given that both systems have a solution.

Note that moreover, if (P) is unbounded, then (D) is infeasible. If (D) is unbounded then (P) is infeasible. On the other hand, it is possible that (P) and (D) are both infeasible.

One observation is that in order to certify optimality for a solution x^* for (P) one can only use constraints in the dual solution y^* that are tight for y^* :

Theorem 6.22 (Complementary slackness). *Let (x^*, y^*) be feasible solutions for*

$$(P) : \max\{c^T x \mid Ax \leq b\} \quad \text{and} \quad (D) : \min\{b^T y \mid A^T y = c; y \geq \mathbf{0}\}$$

Then (x^, y^*) are both optimal if and only if*

$$(A_i^T x^* = b_i \vee y_i^* = 0) \quad \forall i$$

We should note that one can write down LPs in many different forms with min or max objective, equalities or inequalities, non-negativity or not. Each LP will have a dual and its value will coincide with the primal. For convenience we state a very general version from which most desired forms can be easily derived.

$\begin{array}{ll} \max c^T x + d^T u & (P') \\ Ax + Bu \leq a & (\rightarrow y) \\ Cx + Du = b & (\rightarrow z) \\ x \geq \mathbf{0} & \end{array}$	$\begin{array}{ll} \min a^T y + b^T z & (D') \\ A^T y + C^T z \geq c & \\ B^T y + D^T z = d & \\ y \geq \mathbf{0} & \end{array}$
---	---

Theorem 6.23 (Strong Duality II). *One has $(P') = (D')$ assuming both systems are feasible.*

This can be proven by bringing (P') into the inequality form of (P) and then applying Theorem 6.21.

6.8 An application to Nash Equilibria in Zero Sum Games

Finally, we want to study a beautiful application of LP duality. We consider a 2-person game with profit matrices $A, B \in \mathbb{R}^{m \times n}$. A pure strategy of player 1 is to pick a row index $i \in [m]$ and a pure strategy of player 2 is to pick a column index $j \in [n]$. Then the payout for player 1 is $A_{ij} = e_i^T A e_j$ and the payout for player 2 is $B_{ij} = e_i^T B e_j$. The game is called a *zero sum game* if $B = -A$. That means in a zero sum game, the gain of player 1 is the loss of player 2 and the other way around.

Example 6.24. Consider the *rock-paper-scissor* game where the winner gets \$1. Then the profit matrix A is the following

$$A = \begin{array}{c} \begin{array}{ccc} \text{rock} & \text{paper} & \text{scissor} \end{array} \\ \left[\begin{array}{ccc} 0 & -1 & +1 \\ +1 & 0 & -1 \\ -1 & +1 & 0 \end{array} \right] \begin{array}{l} \text{rock} \\ \text{paper} \\ \text{scissor} \end{array} \end{array}$$

Note that rock-paper-scissor is a zero sum game.

Definition 6.25. A pair $(i^*, j^*) \in [m] \times [n]$ is called an *pure Nash equilibrium* if

$$A_{i^*, j^*} \geq \max_{i \in [m]} A_{i, j^*} \quad \text{and} \quad B_{i^*, j^*} \geq \max_{j \in [n]} B_{i^*, j}$$

(in the zero sum case the latter condition is $A_{i^*, j^*} \leq \min_{j \in [n]} A_{i^*, j}$).

In other words, in a pure Nash equilibrium, even knowing the opponents strategy there is no incentive to change ones own strategy. Observe that rock-paper-scissor does *not* have a pure Nash equilibrium.

Now, let us generalize the strategies and allow both players to have *distributions* of the rows and columns. That means the row player may select a so called *mixed strategy* $x \in \mathbb{R}_{\geq 0}^m$ with $\sum_{i=1}^m x_i = 1$ and the column player may select a mixed strategy $y \in \mathbb{R}_{\geq 0}^n$ with $\sum_{j=1}^n y_j = 1$. Then the (expected) payout for player 1 is

$$\mathbb{E}[\text{payout for player 1}] = \sum_{i=1}^m \sum_{j=1}^n \underbrace{\text{Pr}[\text{P1 plays } i, \text{P2 plays } j]}_{x_i \cdot y_j} \cdot A_{ij} = x^T A y$$

Similarly, the (expected) payout for player 2 is $x^T B y$. Let $\Delta_m := \{x \in \mathbb{R}_{\geq 0}^m \mid \sum_{i=1}^m x_i = 1\}$ be the simplex in m dimensions. That means Δ_m is the set of mixed strategies for player 1 and Δ_n are the mixed strategies for player 2.

Definition 6.26. A pair (x^*, y^*) with $x^* \in \Delta_m$ and $y^* \in \Delta_n$ is called a *mixed Nash equilibrium* if

$$(x^*)^T A y^* \geq \max_{x \in \Delta_m} x^T A y^* \quad \text{and} \quad (x^*)^T B y^* \geq \max_{y \in \Delta_n} (x^*)^T B y$$

(again in the zero sum case, the second condition is equivalent to $(x^*)^T A y^* \leq \min_{y \in \Delta_n} (x^*)^T A y$).

Back to the rock-paper-scissor game, we can see that choosing the uniform distribution $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ for both players forms a mixed Nash equilibrium. We can prove that such a mixed Nash equilibrium always exists:

Theorem 6.27. *For any 2-person zero sum game, there is a mixed Nash equilibrium which can be found in polynomial time.*

Proof. We will prove that

$$\max_{x \in \Delta_m} \min_{y \in \Delta_n} x^T A y = \min_{y \in \Delta_n} \max_{x \in \Delta_m} x^T A y \quad (*)$$

then the “outer” x and y will form the mixed Nash equilibrium as for them the order of the max and min does not matter. In order to prove (*) we will express the LHS of (*) as an LP and then apply *LP duality*. First we make a useful observation:

Fact. *Let $c \in \mathbb{R}^n$. Then $\min\{c^T y \mid y \in \Delta_n\}$ is attained by a vertex of the simplex, i.e. $y \in \{e_1, \dots, e_n\}$.*

That means for us that once the opponent has fixed a strategy, a player can w.l.o.g. pick a pure strategy rather than a mixed strategy. We use this to rewrite the LHS of (*) as

$$\max_{x \in \Delta_m} \min_{y \in \Delta_n} x^T A y = \max_{x \in \Delta_m} \min_{j \in [n]} \underbrace{x^T A e_j}_{= A^j} = \max\{t \mid \langle A^j, x \rangle \geq t \forall j \in [n], x \in \Delta_m\} \quad (**)$$

The RHS of (**) is a linear program. For convinience, we write it in standard form and construct the dual:

$\begin{array}{ll} \max t & (P) \\ \sum_{i=1}^m x_i = 1 & (\rightarrow s) \\ t - \sum_{i=1}^m x_i A_{ij} \leq 0 \quad \forall j \in [n] & (\rightarrow y_j) \\ x_i \geq 0 \quad \forall i \in [m] \end{array}$	$\begin{array}{ll} \min s & (D) \\ \sum_{j=1}^n y_j = 1 \quad \forall j \in [n] \\ s - \sum_{j=1}^n y_j A_{ij} \geq 0 \quad \forall i \in [m] \\ y_j \geq 0 \quad \forall j \in [n] \end{array}$
--	--

Then rewriting (D) gives that

$$\min\{s \mid \langle A_i, y \rangle \leq s \forall i \in [m]; y \in \Delta_n\} = \min_{y \in \Delta_n} \max_{i \in [m]} e_i^T A y = \min_{y \in \Delta_n} \max_{x \in \Delta_m} x^T A y$$

□

We would like to briefly comment that swapping the order of \max_x and \min_y can be done in much more generality.

Theorem 6.28 (Sion’s Minimax Theorem). *Let $X \subseteq \mathbb{R}^m$ and $Y \subseteq \mathbb{R}^n$ two compact, convex sets and let $f : X \times Y \rightarrow \mathbb{R}$ be a continuous function so that (i) for each $x \in X$, the function $f(x, \cdot) : Y \rightarrow \mathbb{R}$ is convex and (ii) for each $y \in Y$ the function $f(\cdot, y) : X \rightarrow \mathbb{R}$ is concave. Then*

$$\max_{x \in X} \min_{y \in Y} f(x, y) = \min_{y \in Y} \max_{x \in X} f(x, y)$$

Nash proved more generally that mixed Nash equilibria exist even in games that are not zero sum. In fact, he proved it for an arbitrary finite number of players.

Theorem 6.29 (Nash 1951 [Nas51]). *Let $k \in \mathbb{Z}_{\geq 2}$. Every k -person game (with a finite number of pure strategies per player) has a mixed Nash equilibrium.*

However Nash's proof is based on a fix point theorem and is non-constructive, hence no polynomial time algorithm is known to compute the Nash equilibrium⁹. In fact there is good evidence that there is no efficient algorithm as finding a Nash equilibrium even in a 2-person game can be shown to be **PPAD**-complete.

⁹ The original argument by Nash uses the following result:

Theorem 6.30 (Kakutani 1941). *Let $X \subseteq \mathbb{R}^n$ be non-empty compact convex set and let $\Phi : X \rightarrow 2^X$ be a function satisfying: (i) $\Phi(x) \subseteq X$ is a non-empty convex set for all $x \in X$; (ii) the graph $\{(x, y) : x \in X, y \in \Phi(x)\}$ is closed. Then Φ has a fix-point, i.e. there is an $x^* \in X$ so that $x^* \in \Phi(x^*)$.*

Now to Nash's argument: consider k players and for the sake of simplicity assume each player has m pure strategies denoted by the standard basis vectors e_1, \dots, e_m . If $x := (x^{(1)}, \dots, x^{(k)})$ with $x^{(i)} \in \{e_1, \dots, e_m\}$ is a vector of pure strategies chosen by the players, then we denote $A_i(x)$ as the payout for player i . Note that Δ_m is then the set of mixed strategies for each player. For $x \in X := \Delta_m^k$ being a vector of mixed strategies we also write $A_i(x)$ as the (expected) payout for player i if each player independently picks a pure strategy according to x . We note that the set X is non-empty, compact (= bounded + closed) and convex. We also write $BR_i(x) := \{(x^{(1)}, \dots, x^{(i-1)}, y, x^{(i+1)}, \dots, x^{(k)}) \mid A_i(x^{(1)}, \dots, x^{(i-1)}, y, x^{(i+1)}, \dots, x^{(k)}) \geq A_i(x^{(1)}, \dots, x^{(i-1)}, z, x^{(i+1)}, \dots, x^{(k)}) \forall z \in \Delta_m\}$ as the set of best responses of player i assuming the other players keep their strategies as in x . Then for each x , the set $BR_i(x)$ is convex. Next, we define a map $\Phi : X \rightarrow 2^X$ as follows: for $x = (x^{(1)}, \dots, x^{(k)}) \in X$ define $\Phi(x) := BR_1(x) \times \dots \times BR_k(x)$ as the set of best responses that the players have. Then $\Phi(x)$ is a non-empty convex set. We leave it to the reader to verify that the graph of Φ is indeed closed. Hence Kakutani's Theorem applies and there is a fix point x^* with $x^* \in \Phi(x^*)$. That means for each player i , the strategy $(x^*)^{(i)}$ is already a best response to x^* . Hence x^* is a Nash equilibrium.

Chapter 7

Submodular functions

Numerous optimization problems from machine learning over approximation algorithms to combinatorial optimization share a *diminishing returns* property. Instead of problem-specific approaches one can capture many such problems with the framework of *submodular functions*.

Definition 7.1. Let $n \in \mathbb{N}$. A function $f : 2^{[n]} \rightarrow \mathbb{R}$ is called *submodular* if

$$f(A \cup \{j\}) - f(A) \geq f(B \cup \{j\}) - f(B) \quad \forall A \subseteq B \subseteq [n] \quad \forall j \in [n] \setminus B$$

In other words: a set function is submodular if adding a new element j to a smaller set A causes greater (or equal) increase than adding it to a larger set B . Additional desirable properties or subclasses of submodular functions are:

- *Non-negativity*, i.e. $f(S) \geq 0 \quad \forall S \subseteq [n]$
- *Monotonicity*, i.e. $f(A) \leq f(B) \quad \forall A \subseteq B \subseteq [n]$
- *Symmetry*, i.e. $f(S) = f([n] \setminus S) \quad \forall S \subseteq [n]$

We want to give a two examples of submodular functions.

- **Coverage functions.** Let $S_1, \dots, S_n \subseteq U$ be a *set family* over a *ground set* U . Then the number of covered elements given by $f(I) := |\bigcup_{i \in I} S_i|$ is a monotone submodular function. Similarly this holds for the volume of sets and the probability of the union of events.
- **Cut functions in graphs.** Let $G = (V, E)$ be an undirected graph with edge weights $w : E \rightarrow \mathbb{R}_{\geq 0}$. Then the function $f(S) := w(\delta(S)) = \sum_{\{i,j\} \in E: \{i,j\} \cap S = \{i\}} w(i,j)$ giving the value of the cut S is a symmetric submodular function (though it is not monotone!). Similar for directed graphs $D = (V, A)$, the value $w(\delta^+(S))$ of a directed cut is submodular (though neither symmetric nor monotone).

Often one can find an alternative definition of submodularity in the literature:

Lemma 7.2 (Equivalent Characterization of Submodularity). *Let $f : 2^{[n]} \rightarrow \mathbb{R}$. Then the following is equivalent*

- (i) f is submodular.
- (ii) One has $f(A) + f(B) \geq f(A \cap B) + f(A \cup B) \quad \forall A, B \subseteq [n]$

We also make an observation that is often useful:

Lemma 7.3. *Let $f, g : 2^{[n]} \rightarrow \mathbb{R}$ be two submodular functions and let $\lambda \geq 0$. Then*

- (i) $f + g$ is submodular
- (ii) λf is submodular

For algorithms dealing with submodular functions, typically the assumption is that one has *oracle access* to the function f , that means the algorithms only need to be able to evaluate the function value $f(S)$ for any given set S . Here provide a few comments over known results:

- The *MinCut* problem is a special case of *minimizing a submodular function* and we have seen in Chapter 1 that this special case can be done in polynomial time. In fact, for any submodular function one can solve the problem $\min\{f(S) : S \subseteq V\}$ in polynomial time [GLS88]. The algorithm is based on the ellipsoid method and is beyond the scope of this class. We recommend Chapter 44 of Schrijver [Sch03] for a readable exposition.
- The *MaxCut* problem is a special case of *maximizing a non-negative submodular function* which implies that the latter problem must be NP-hard. One can prove that the latter problem has a polynomial time $\frac{1}{2}$ approximation [BFNS15]. One can also prove that finding a $(\frac{1}{2} + \epsilon)$ -approximation would require an exponential number of queries to f [FMV11].

7.1 Maximizing a monotone submodular functions with a cardinality constraint

In this section, we present a classical result due to Nemhauser, Wolsey, Fisher [NWF78] while we follow the exposition of the recent survey of Buchbinder and Feldman [BF18]. We study the following problem of *maximizing a submodular function subject to*

7.1. MAXIMIZING A MONOTONE SUBMODULAR FUNCTIONS WITH A CARDINALITY CONSTRAINT

a cardinality constraint: Given oracle access to a non-negative monotone submodular function $f : 2^{[n]} \rightarrow \mathbb{R}_{\geq 0}$ and a parameter $k \in \mathbb{N}$. The goal is to solve (or approximate)

$$\max\{f(S) : S \subseteq [n] \text{ and } |S| \leq k\}$$

Without the cardinality constraint $|S| \leq k$, the optimum would simply be attained by $S = [n]$. Note that this problem already captures the maximum set coverage problem $\max\{|\bigcup_{i \in I} S_i| : |I| \leq k\}$ which is known to be **NP**-hard to approximate within a factor of $1 - \frac{1}{e} - \varepsilon$ for any $\varepsilon > 0$. We denote $f(A | B) := f(A \cup B) - f(B)$ as the *marginal increase* when adding A to B . In particular for single elements $i \in [n]$ we will write $f(i | A) = f(A \cup \{i\}) - f(A)$. We consider the following simple algorithm:

The Submodular Greedy Algorithm

Input: A monotone submodular function $f : 2^{[n]} \rightarrow \mathbb{R}_{\geq 0}$ and $k \in \mathbb{Z}_{\geq 0}$

Output: A set $(1 - \frac{1}{e})$ -approximation to $\max\{f(S) \mid |S| = k\}$.

- (1) Set $S_0 := \emptyset$
- (2) FOR $i = 1$ TO k DO
 - (3) Let $x_i \in [n] \setminus S_{i-1}$ be the element maximizing $f(x_i | S_{i-1})$
 - (4) Set $S_i := S_{i-1} \cup \{x_i\}$
- (5) Return S_k

We begin with useful inequality which says that the gain of a whole set B must be at least as the gain of all its elements:

Lemma 7.4. *Let $f : 2^{[n]} \rightarrow \mathbb{R}$ be a submodular function. Then for any $A, B \subseteq [n]$ one has $\sum_{i \in B} f(i | A) \geq f(B | A)$.*

Proof. W.l.o.g. suppose $B = \{1, \dots, \ell\}$. Then

$$f(B | A) = \sum_{i=1}^{\ell} f(i | A \cup \{1, \dots, i-1\}) \stackrel{f \text{ submod.}}{\leq} \sum_{i=1}^{\ell} f(i | A).$$

□

Now we can analyze the algorithm:

Theorem 7.5 (Nemhauser, Wolsey, Fisher [NWF78]). *The greedy algorithm gives a $(1 - \frac{1}{e})$ -approximation for maximizing a non-negative monotone submodular function subject to a cardinality constraint.*

Proof. Let $S^* := \operatorname{argmax}\{f(S) \mid S \subseteq [n], |S| = k\}$ be the optimum solution. We will prove that $f(S_k) \geq (1 - \frac{1}{e})f(S^*)$ where S_k is the last iterate in the algorithm. Let $i \in \{1, \dots, k\}$. The crucial argument is that the *marginal increase* $f(x_i \mid S_{i-1})$ is at least as large as if we were adding an *average* element from S^* . More precisely,

$$\begin{aligned} f(S_i) - f(S_{i-1}) &= f(x_i \mid S_{i-1}) \\ &\geq \mathbb{E}_{x \sim S^*} [f(x \mid S_{i-1})] \\ &\stackrel{f \text{ subm.} + \text{Lem 7.4}}{\geq} \frac{1}{k} (f(S^* \cup S_{i-1}) - f(S_{i-1})) \\ &\stackrel{f \text{ monotone}}{\geq} \frac{1}{k} (f(S^*) - f(S_{i-1})) \end{aligned}$$

In other words, in every iteration we close at least a $\frac{1}{k}$ -fraction of the remaining gap to the optimum value. This can be rearranged to

$$f(S^*) - f(S_i) \leq \left(1 - \frac{1}{k}\right) \cdot (f(S^*) - f(S_{i-1})).$$

Iterating this inequality k times gives

$$f(S^*) - f(S_k) \leq \underbrace{\left(1 - \frac{1}{k}\right)^k}_{\leq 1/e} \cdot \underbrace{(f(S^*) - f(S_0))}_{\geq 0} \leq \frac{1}{e} \cdot f(S^*)$$

which means that $f(S_k) \geq (1 - \frac{1}{e}) \cdot f(S^*)$ as claimed. \square

We would also like to note that the greedy algorithm makes at most nk queries to the function f .

7.2 Application to the Target Set Selection Problem

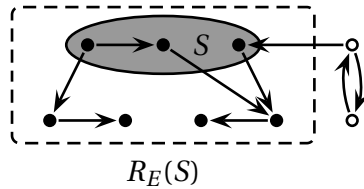
When working on a problem it is extremely useful if one is able to realize that the problem has indeed a submodular behavior. We demonstrate this with an application that is due to Kempe, Kleinberg and Tardos [KKT05].

Suppose you work for a company that has a new product. In order to market it you want to give it out for free to a certain number of influencers to stimulate others to buy the product. We model this as follows: We have a population of n and we have a matrix $P = (P_{ij})_{ij} \in [0, 1]^{n \times n}$ of probabilities with the interpretation that if i has the new product then with probability P_{ij} person j will be influenced to buy the product as well (if j does not already have it). We assume that the events/decisions are independent. Moreover we have a budget of k units that

we want to give out for free. The objective is to select a subset $S \subseteq [n]$ of $|S| \leq k$ persons so that the expected number of persons that acquire¹ the product is maximized. We denote that expected number by $f(S)$. We claim that

Lemma 7.6. *The objective $f : 2^{[n]} \rightarrow \mathbb{R}_{\geq 0}$ is a monotone submodular function.*

Proof. We write $E \sim P$ to indicate that E is a set of independently and randomly sampled directed edges where (i, j) is included with probability P_{ij} . We note that a person $i \in [n] \setminus S$ will buy the new product if and only if there is a directed path from S to i in the directed graph $([n], E)$. For edges E and $S \subseteq [n]$ we write $R_E(S)$ as the subset of vertices of the graph $([n], E)$ that are reachable from S .



Then

$$f(S) = \mathbb{E}_{E \sim P} [|R_E(S)|] = \mathbb{E}_{E \sim P} \left[\underbrace{\left| \bigcup_{i \in S} R_E(i) \right|}_{=: f_E(S)} \right] \quad (7.1)$$

We note that the inner function f_E is a *coverage function* which we know is submodular. Then f is the *average* of coverage functions and hence by Lemma 7.3, f is also submodular. From (*) we see that f is monotone (because $R_E(S)$ is monotone). \square

So we could apply the algorithm from Section 7.1 to find a $(1 - \frac{1}{e})$ -approximation. The only technicality is that we would need to be able to evaluate $f(S)$ for every S . There does not seem to be a good direct way to compute $f(S)$. But using the interpretation from Eq (7.1) we know that we can use repeated sampling to estimate $f(S)$ (see Section 1.7), because for a fixed S and E we can compute $|R_E(S)|$ in polynomial time using e.g. breadth-first-search..

¹I.e. either get it for free or buy it.

Bibliography

- [AMS96] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *STOC*, pages 20–29. ACM, 1996.
- [ARV09] Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. *J. ACM*, 56(2), apr 2009.
- [Bal97] Keith Ball. An elementary introduction to modern convex geometry. In *Flavors of geometry*, volume 31 of *Math. Sci. Res. Inst. Publ.*, pages 1–58. Cambridge Univ. Press, Cambridge, 1997.
- [BF18] Niv Buchbinder and Moran Feldman. Submodular functions maximization problems. In Teofilo F. Gonzalez, editor, *Handbook of Approximation Algorithms and Metaheuristics, Second Edition, Volume 1: Methodologies and Traditional Applications*, pages 753–788. Chapman and Hall/CRC, 2018.
- [BFNS15] Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. A tight linear time $(1/2)$ -approximation for unconstrained submodular maximization. *SIAM J. Comput.*, 44(5):1384–1402, 2015.
- [BvH82] Allan Borodin, Joachim von zur Gathen, and John Hopcroft. Fast parallel matrix and gcd computations. *Information and Control*, 52(3):241–256, 1982.
- [Chr76] Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. *Operations Research Forum*, 3, 1976.
- [CW13] Kenneth L. Clarkson and David P. Woodruff. Low rank approximation and regression in input sparsity time. In *STOC*, pages 81–90. ACM, 2013.
- [Dan90] George B. Dantzig. *Origins of the simplex method*, pages 141–151. Association for Computing Machinery, New York, NY, USA, 1990.

- [Edm65] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.
- [Fei98] Uriel Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45(4):634–652, jul 1998.
- [FKS84] Michael L. Fredman, János Komlós, and Endre Szemerédi. Storing a sparse table with $O(1)$ worst case access time. *J. ACM*, 31(3):538–544, jun 1984.
- [FMV11] Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.
- [GLS88] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2. 1988.
- [GW95] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.
- [HJ90] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 1990.
- [Kal92] Gil Kalai. A subexponential randomized simplex algorithm (extended abstract). In *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing*, STOC '92, pages 475–482, New York, NY, USA, 1992. Association for Computing Machinery.
- [Kar84] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, STOC '84, pages 302–311, New York, NY, USA, 1984. Association for Computing Machinery.
- [Kar93] David R. Karger. Global min-cuts in rnc, and other ramifications of a simple min-cut algorithm. In *SODA*, pages 21–30. ACM/SIAM, 1993.
- [Kha79] L. G. Khachyan. A polynomial algorithm in linear programming. *Dokl. Akad. Nauk SSSR*, 244(5):1093–1096, 1979.
- [Kho02] Subhash Khot. On the power of unique 2-prover 1-round games. In *STOC*, pages 767–775. ACM, 2002.

- [KKG21] Anna R. Karlin, Nathan Klein, and Shayan Oveis Gharan. A (slightly) improved approximation algorithm for metric TSP. In *STOC*, pages 32–45. ACM, 2021.
- [KKG22] Anna R. Karlin, Nathan Klein, and Shayan Oveis Gharan. A (slightly) improved bound on the integrality gap of the subtour LP for TSP. In *FOCS*, pages 832–843. IEEE, 2022.
- [KKMO07] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O’Donnell. Optimal inapproximability results for max-cut and other 2-variable csp’s? *SIAM Journal on Computing*, 37(1):319–357, 2007.
- [KKT05] David Kempe, Jon M. Kleinberg, and Éva Tardos. Influential nodes in a diffusion model for social networks. In *ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 1127–1138. Springer, 2005.
- [KR08] Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within $2 - \hat{\mu}$. *Journal of Computer and System Sciences*, 74(3):335–349, 2008. Computational Complexity 2003.
- [KS93] David R. Karger and Clifford Stein. An $o(n^2)$ algorithm for minimum cuts. In *STOC*, pages 757–765. ACM, 1993.
- [KV12] B. H. Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer-Verlag, New York, NY, 2012.
- [LR99] Tom Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, nov 1999.
- [McS01] F. McSherry. Spectral partitioning of random graphs. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 529–537, 2001.
- [Mul86] Ketan Mulmuley. A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. In *STOC*, pages 338–339. ACM, 1986.
- [MV80] Silvio Micali and Vijay V. Vazirani. An $o(v|v|c|e|)$ algorithm for finding maximum matching in general graphs. In *21st Annual Symposium on Foundations of Computer Science (sfcs 1980)*, pages 17–27, 1980.
- [MVV87] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

- [Nas51] John Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951.
- [NWF78] George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions - I. *Math. Program.*, 14(1):265–294, 1978.
- [OGT15] Shayan Oveis Gharan and Luca Trevisan. A new regularity lemma and faster approximation algorithms for low threshold rank graphs. *Theory of Computing*, 11(9):241–256, 2015.
- [Sch03] A. Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer, 2003.
- [Spe85] Joel Spencer. Six standard deviations suffice. *Transactions of the American Mathematical Society*, 289(2):679–706, 1985.
- [ST04] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM*, 51(3):385–463, 2004.
- [Tut47] W. T. Tutte. The factorization of linear graphs. *Journal of the London Mathematical Society*, s1-22(2):107–111, 1947.
- [Vaz20] Vijay V. Vazirani. A proof of the MV matching algorithm. *CoRR*, abs/2012.03582, 2020.
- [Vu14] Van Vu. A simple svd algorithm for finding hidden partitions, 2014.

Appendix A

Notation and useful facts

Here we list some of the notation and auxiliary results that have been used throughout these notes to have them in one place.

General notation. For $n \in \mathbb{N}$ we write $[n] = \{1, \dots, n\}$. For $S \subseteq [n]$ we denote $\mathbf{1}_S$ as the *characteristic vector* of S , i.e. $\mathbf{1}_S \in \{0, 1\}^n$ is a vector with entries $\mathbf{1}_S(i) = 1$ if and only if $i \in S$.

Functions. Let A and B two sets. A function $f : A \rightarrow B$ is called *injective* if for all $a, a' \in A$ with $a \neq a'$ one has $f(a) \neq f(a')$. The function f is called *surjective* if for all $b \in B$ there is an $a \in A$ with $f(a) = b$. The function f is *bijective* if it is both injective and surjective. In case that A and B are finite sets and f is bijective we know that $|A| = |B|$. For $b \in B$ we write $f^{-1}(b) := \{a \in A : f(a) = b\}$. For a subset $A' \subseteq A$ we write $f|_{A'}$ as the *restriction* of f to A' . Then $f|_{A'} : A' \rightarrow B$ with $f|_{A'}(a) = f(a)$ for all $a \in A'$.

Matrices and vectors. For a matrix $A \in \mathbb{R}^{m \times n}$ we denote the columns as $A^1, \dots, A^n \in \mathbb{R}^m$ and the row vectors as $A_1, \dots, A_m \in \mathbb{R}^n$. The entry at position (i, j) is denoted as A_{ij} . For vectors $x, y \in \mathbb{R}^n$ we write the standard inner product as $\langle x, y \rangle := \sum_{i=1}^n x_i y_i$. By convention, for us a vector is always a column vector.

Norms and geometry. For a vector $x \in \mathbb{R}^n$ and $1 \leq p < \infty$ one defines the ℓ_p -norm as $\|x\|_p := (\sum_{i=1}^n |x_i|^p)^{1/p}$, though in these notes we have only used the cases $p \in \{1, 2\}$ where are $\|x\|_1 = \sum_{i=1}^n |x_i|$ and $\|x\|_2 = (\sum_{i=1}^n |x_i|^2)^{1/2}$. The case $p = \infty$ is defined as $\|x\|_\infty = \max\{|x_i| : i = 1, \dots, n\}$.

Theorem A.1 (Cauchy-Schwarz Inequality I). *For any $x, y \in \mathbb{R}^n$ one has $|\langle x, y \rangle| \leq \|x\|_2 \cdot \|y\|_2$.*

This inequality represents the fact that the inner product between two vectors is maximized when they are colinear. In fact, the inequality can be generalized to certain pairs of ℓ_p -norms. We state one useful case:

Theorem A.2 (Cauchy-Schwarz Inequality II). *For any $x, y \in \mathbb{R}^n$ one has $|\langle x, y \rangle| \leq \|x\|_1 \cdot \|y\|_\infty$.*

Another useful fact is that for $x \in \mathbb{R}^n$ one has $\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n}\|x\|_\infty$ as well as $\|x\|_2 \leq \|x\|_1 \leq \sqrt{n}\|x\|_2$.

Probability theory. If S is a set then we write $X \sim S$ to indicate that X is a random variable that is chosen uniformly from S . That means if S is finite, then $\Pr[X = i] = \frac{1}{|S|}$ for all $i \in S$.