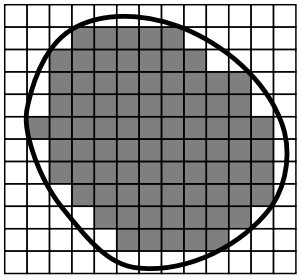
Computational geometry: notes 1*

Topics covered in this course

In this course we shall mainly cover two topics in computational geometry. The first is computation volumes of convex bodies, and the second is the geometric range searching.

Given a convex body[1] C in $\mathbb{R}^d$, the task is to find its volume. If d is small, then the conventional method of subdividing the space into tiny boxes and counting the number of boxes that are contained in C might be acceptably fast. If n is large, then the number of boxes is exponential in d, and the method is utterly impractical.



Do not do it in the 4-space!

We shall see a method to approximate the volume of C which takes only polynomially many steps. The basic idea is simple: place a ball B inside C, and pick points at random from C. Then the proportion of points that falls into B gives a good estimate on the volume of C. To make this work we shall need ellipsoidal method, isoperimetric inequalities, and some results on mixing of random walks.

Secondly, we shall study the geometric range searching. A typical problem is this: Suppose we have a collection P of points in the plane, we wish to be able to find all the points in P that fall within a given disk D. If we need to solve this problem just once, then we are forced to perform $\Omega(|P|)$ obvious checks. However, if the set P is fixed, and we need to solve the problem many times for different D, there is room for cleverness.

About computational model

We focus on the underlying mathematics rather than on the practical side of computing. Thus instead of merely sweeping the details about real-world computers under the rug, we will throw them outside.

We shall assume that we compute with genuine real numbers, and not with approximations that can be represented on a computer. In practice the danger

*These notes are from `http://www.borisbukh.org/CompGeomEaster11/notes1.pdf`.

[1] *Convex body* is a closed and bounded convex set.

of round-off errors is not only in their potential to accumulate, but also in the loss many mathematical properties. For example, in machine arithmetic the associative law $x + (y + z) = (x + y) + z$ need not hold. Dealing with these problems is an art in itself.

We shall ignore the multiplicative constants in the estimating the cost of the computation. For example, both an algorithm that takes n^2 steps, and the algorithm that takes $100n^2$ steps for us will be simply an $\Theta(n^2)$ algorithm. An algorithm that takes $\Theta(n \log n)$ steps will be considered superior to one that takes $\Theta(n^2)$. This is correct for most simple algorithms, and is wrong for complicated algorithms because the latter usually come with large implicit constants.

Finally, we shall usually make non-degeneracy assumptions on the input to the algorithm. These are made to simplify the exposition, and will always be explicitly stated.

Basic definitions and standard facts

A *convex combination* of two points $p, q \in \mathbb{R}^d$ is any point of the form $\lambda p + (1-\lambda)q$ where $0 \leq \lambda \leq 1$, i.e. any point on the line segment pq. A set $C \subset \mathbb{R}^d$ is a *convex set* if for every $p, q \in C$ it contains the segment pq. The definition The *convex hull* of a set $P \subset \mathbb{R}^d$ is the smallest convex set containing P,

$$\operatorname{conv}(P) = \bigcap_{\substack{\text{convex} \\ P \subset C}} C.$$

Computing volumes: oracles

There are different ways a convex body C might be presented to us. For example, the C might be given as the intersection of halfspaces, or ellipsoids. It could stored in computer's memory, or parts of it could be computed as needed. We are not interested in these details, and so assume that we have an efficient method to determine whether a given point is in the body. We personify the method, and call it an *oracle*. An oracle is simply an entity that answers questions of a particular form. For example, the *strong membership oracle* answers "Is the point ___ in C?". The type of oracle we will use most is the *strong separation oracle* which answers "Is the point ___ in C? If no, what is the hyperplane separating it from C?".

If when presented to every point we choose, the oracle responds "no", it does not mean that C is empty, but simply that C might be elsewhere. To avoid this we shall assume that we know of a ball B_{in} contained in C. Similarly, to avoid the situation where oracle responds "yes" all the time, we shall assume that we know of a ball B_{out} containing C.

We shall assume that our algorithms can access the convex body only via an oracle, and not in any direct way.

Computing volumes: deterministic algorithms

Most familiar computational paradigm is the deterministic algorithm. Such an algorithm performs a sequence of steps that depends only on the input presented to it. Alas, no deterministic algorithm can quickly approximate the volume of a convex body quickly. The reason is that if P_0 and P_1 are the sets of points to which the oracle replied "yes" and "no", then the only conclusion one can draw is that $\operatorname{vol}\operatorname{conv}(P_0) \le \operatorname{vol} C < \operatorname{vol}\operatorname{conv}(P_1)$, where $\operatorname{conv}(P)$ denotes the convex hull of P.

Theorem 1 (Elekes)**.** *If $P \subset \mathbb{R}^d$ is an n-point set contained in the unit ball $B(0,1) \in \mathbb{R}^d$, then*

$$\frac{\operatorname{vol}\operatorname{conv}(P)}{\operatorname{vol} B(0,1)} \le n/2^d.$$

For a point $p \in \mathbb{R}^d$ let $B_p \stackrel{\text{def}}{=} B(p/2, |p|/2)$ denote the smallest ball containing both p and the origin. The theorem 1 is a consequence of the following geometric lemma.

Lemma 2. *Suppose $p, q \in \mathbb{R}^d$ are arbitrary points, and $r = \lambda p + (1-\lambda)q$ is their convex combination. Then $B_r \subset B_p \cup B_q$.*

In particular, if P is any set in $\mathbb{R}^d$, then

$$\operatorname{conv} P \subset \bigcup_{p \in P} B_p.$$

Proof. A point x is contained in B_p precisely when $\langle x - p/2, x - p/2\rangle \le \langle p/2, p/2\rangle$. After expanding we obtain

$$x \in B_p \iff \langle x, x\rangle \le \langle p, x\rangle.$$

Similarly,

$$\begin{aligned} y \in B_q &\iff \langle y, y\rangle \le \langle q, y\rangle, \\ y \in B_r &\iff \langle y, y\rangle \le \langle r, y\rangle \\ &\iff \langle y, y\rangle \le \lambda\langle p, y\rangle + (1-\lambda)\langle q, y\rangle. \end{aligned}$$

The inclusion $B_r \subset B_p \cup B_q$ is then clear.

Since every element of $\operatorname{conv}(P)$ is can be obtained from points of P by repeatedly taking convex combinations, the lemma follows. □

Proof of theorem 1. From the preceding lemma we have $\operatorname{conv}(P) \subset \bigcup_{p \in P} B_p$, and thus

$$\operatorname{vol}\operatorname{conv}(P) \le \sum \operatorname{vol} B_p = \operatorname{vol} B(0,1) \sum |p|^d \le 2^{-d} B(0,1).$$

□

Theorem 3. *There is no deterministic algorithm that makes fewer than 2^{d-2} queries to a strong separation oracle, and outputs a number V such that $\frac{1}{2}V \leq \operatorname{vol} C \leq 2V$.*

Proof. If the oracle answers consistently with $C = B(0,1)$, then after asking 2^{d-2} questions, the $volC$ can be any number between $\frac{1}{4}\operatorname{vol} B(0,1)$ and $\operatorname{vol} B(0,1)$. □

Computing volumes: basic probabilistic algorithm

The oracles from theorem 1 are like their namesakes from antiquity. Their answers contain little information, and are consistent with many interpretations. However, the computational oracles are not deceptive, and truthfully answer queries about a single convex body C. We shall exploit that.

The probabilistic algorithm is allowed to decide what to do next based on a result of a random experiment, such as a coin toss. The simplest probabilistic algorithm to compute $\operatorname{vol} C$ is this:

Algorithm 1 Darts algorithm

```
1: procedure DARTS(B_in, B_out, δ)      ▷ Computes vol C with relative error δ
2:     N ← 10/δ²
3:     Throws ← 0
4:     Hits ← 0
5:     while Hits < N do
6:         p ← a random point from B_out
7:         Throws ← Throws + 1
8:         if p ∈ C then                                  ▷ Use oracle for this
9:             Hits ← Hits + 1
10:        end if
11:    end while
12:    return (Hits/Throws) vol B_out
13: end procedure
```

The algorithm simply throws imaginary darts into "dartboard" $B(0,1)$ until it hits the "target" C exactly N times. The algorithm uses the proportion of hits to estimate the volume of C. We say that the *relative error* of the algorithm is at most δ if it output a number V such that $(1-\delta)V \leq \operatorname{vol} C \leq (1+\delta)V$. Since we want to make the error small, and to avoid unnecessary calculations we shall always assume that δ is at most $1/2$.

Theorem 4. *The probability that the algorithm makes more than $10N \operatorname{vol} B_{out}/\operatorname{vol} C$ oracle calls is at most $1/10$. The algorithm errs with relative error more than δ with probability at most $1/10$.*

Proof. Let X_i be the number of throws the algorithm makes after i'th hit before obtaining $i+1$'st hit (we count the throw scoring $i+1$'st hit). The variables

$X_1, \ldots, X_N$ are independent, and each of them is distributed according to geometric distribution with success probability $p = \operatorname{vol} C / \operatorname{vol} B_{\text{out}}$. Furthermore, $Throws = X_1 + \cdots + X_N$. Thus, the expectation and variance of $Throws$ are

$$\begin{aligned}\mathbf{E}[Throws] &= N/p, \\ \mathbf{V}[Throws] &= N(1-p)/p^2.\end{aligned}$$

By Markov inequality the probability that the algorithm makes more than $10N/p$ oracle queries is at most $\mathbf{E}[Throws]/(10N/p) = 1/10$. Since $p \geq \operatorname{vol} B_{\text{in}} / \operatorname{vol} B_{\text{out}}$, the first claim is true.

The probability that the relative error is at most δ is

$$\begin{aligned}\mathbf{P}\left[\left|\frac{Hits}{Throws}\operatorname{vol} B_{\text{out}} - \operatorname{vol} C\right| < \delta \operatorname{vol} C\right] &= \mathbf{P}\left[\left|\frac{N}{p\ Throws} - 1\right| < \delta\right] \\ &= \mathbf{P}[\frac{\delta}{1+\delta} < 1 - \frac{Throws}{N/p} < \frac{\delta}{1-\delta}] \\ &\geq \mathbf{P}[|N/p - Throws| < \delta N/p]\end{aligned}$$

which by Chebyshev's inequality is

$$\begin{aligned}&\geq 1 - \frac{N(1-p)/p^2}{(\delta N/p)^2} \\ &= 1 - \frac{1-p}{\delta^2 N} \geq 9/10.\end{aligned}$$

□

The error probability of 1/10 in the algorithm can clearly be reduced by tweaking the algorithm, and its analysis. Whereas it is easy to do it for this simple algorithm, we will not want to do later, as that introduces extra calculations in the algorithm analysis. The conceptually simpler way is to abort the algorithm whenever it takes more than $10N \operatorname{vol} B_{\text{out}} / \operatorname{vol} B_{\text{in}}$ oracle calls. Then we can run this modified algorithm $\Omega(\log(1/\epsilon))$ times, and output the median of all the results computed. One can show that with probability $1 - \epsilon$ the median approximates $\operatorname{vol} C$ with relative error at most δ.

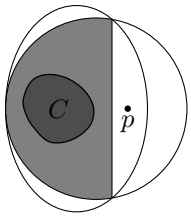
Computing volumes: ellipsoid method

The running time of DARTS algorithm is proportional to the ratio $\operatorname{vol} B_{\text{out}} / \operatorname{vol} C$. The ratio might actually be arbitrarily large, for unlucky choice of B_{out}. The ellipsoid algorithm is way to improve on a choice of B_{out}. It uses the strong separation oracle. An *ellipsoid* is an image of a ball under an affine map.

Algorithm 2 Ellipsoid algorithm

1: **procedure** EllipsoidMethod($B_{\text{in}}, B_{\text{out}}$) ▷ Finds a pair of similarly sized ellipsoids $B'_{\text{in}}, B'_{\text{out}}$ such that $B'_{\text{in}} \subset C \subset B'_{\text{out}}$
2: $B'_{\text{in}} \leftarrow B_{\text{in}}$
3: $B'_{\text{out}} \leftarrow B_{\text{out}}$
4: **while** B'_{out} is bigger than B'_{in} **do**
5: Make an affine change of coordinates so that $B'_{\text{out}} = B(0,1)$
6: $p_0, \ldots, p_d \leftarrow$ vertices of a regular simplex with $|p_i| = 1/8d$
7: **if** $p_i \in C$ for all $i = 0, 1, \ldots, d$ **then**
8: $B'_{\text{in}} \leftarrow B(0, 1/8d^2)$
9: **return** $(B'_{\text{in}}, B'_{\text{out}})$
10: **end if**
11: $H \leftarrow$ the halfspace containing C, but not $p_i \notin C$
12: $B'_{\text{out}} \leftarrow$ the smallest ellipsoid containing $B'_{\text{out}} \cap H$.
13: **end while**
14: **end procedure**

To keep the algorithm readable we assume that the algorithm keeps track of matrix describing the affine change of coordinates, so that in the current coordinates we always have $B'_{\text{out}} = B(0,1)$. At the each iteration, the algorithm replaces the ellipsoid B'_{out} by a smaller ellipsoid which still contains C. It does so by checking for several points p_i lying very close to the center of B'_{out} whether they lie in C or not. If one of them is not in C, then there is a hyperplane separating it from C, and we use that hyperplane to cut B'_{out} to obtain a truncated ellipsoid as in the picture on the right. The truncated ellipsoid can then be enclosed in an ellipsoid of smaller volume than B'_{out}. If all p_i are in C, then their convex hull contains a fairly large ellipsoid. The last statement is elementary, and is the content of the next lemma.



Shrinking B'_{out}

Lemma 5. *If $\sigma \subset \mathbb{R}^d$ is the regular simplex centered at the origin, all of whose vertices are on the unit sphere, then $B(0, 1/d) \subset \sigma$.*

Proof. Let p be any vertex of σ, and let τ be the face opposite to p. The line $0p$ is normal to τ, and meet τ in its barycenter $B(\tau)$. Thus $0p$ is a height of the simplex σ with base τ. It is also the height of the simplex $\operatorname{conv}(\tau \cup \{0\})$ with the same base. We thus obtain

$$\frac{\operatorname{vol} S}{\operatorname{vol} \operatorname{conv}(\tau \cup \{0\})} = \frac{1 + |B(\tau)|}{|B(\tau)|}.$$

Since S is union of $d+1$ simplices, each of which is congruent to $\operatorname{conv}(\tau \cup \{0\})$, the left side is equal to $d+1$. Solving the equation we obtain $|B(\tau)| = 1/d$. □

Since the algorithm terminates only if it reaches line 9, the proceeding lemma guarantees that output of the algorithm is always correct, i.e. $B'_{\text{in}} \subset C \subset B'_{\text{out}}$, and the ratio $\operatorname{vol} B'_{\text{out}} / \operatorname{vol} B'_{\text{in}}$ is $1/(8d^2)^d$. However, does the algorithm ever

terminate? It does and it does it fast since the volumes of B'_{out} decrease rapidly at each iteration.

Lemma 6. *Suppose $H \subset \mathbb{R}^d$ is a halfspace such that the distance from the origin to ∂H is at most $1/8d$. Then $B(0,1) \cap H$ is contained in an ellipsoid of volume at most $(1 - 1/40d) \operatorname{vol} B(0,1)$.*

Proof. Without loss of generality H is of the form $H = \{x_d \le 1/8d\}$. For sake of brevity denote the projection of a vector $x = (x_1, \ldots, x_{d-1}, x_d)$ onto the first $d-1$ coordinates by $x' \stackrel{\text{def}}{=} (x_1, \ldots, x_{d-1})$. Then the ellipsoid

$$E = \{x : a(x_d + \Delta)^2 + b|x'|^2 \le 1\},$$

where

$$a = 1 + \frac{1}{8d} \qquad b = 1 - \frac{1}{16d^2} \qquad \Delta = \frac{1}{8d}.$$

contains $H \cap B(0,1)$. Indeed, if $x \in B(0,1) \cap H$ then

$$\begin{aligned}
a(x_d + \Delta)^2 + b|x'|^2 &= (a-b)x_d^2 + 2a\Delta x_d + a\Delta^2 + b(x_d^2 + |x'|^2) \\
&\le (a-b)x_d^2 + 2a\Delta x_d + a\Delta^2 + b \\
&\le \begin{cases} (b - a + 2a\Delta)x_d + b + a\Delta^2 & \text{if } x_d \in [-1, 0] \\ (a-b)/(8d)^2 + b + 2a\Delta/8d + a\Delta^2 & \text{if } x_d \in [0, 1/8d] \end{cases} \\
&\le \begin{cases} a - 2a\Delta + a\Delta^2 & \text{if } x_d \in [-1, 0] \\ b + 1/128d^2 + a\Delta/4d + a\Delta^2 & \text{if } x_d \in [0, 1/8d] \end{cases}
\end{aligned}$$

and substituting values of a, b, Δ we obtain

$$\begin{aligned}
&\le \begin{cases} 1 + \frac{1}{8d} - \frac{2a}{8d} + \frac{2}{8d} & \text{if } x_d \in [-1, 0] \\ 1 - \frac{1}{16d^2} + \frac{1}{128d^2} + \frac{1+1/8}{8d \cdot 4d} + \frac{1+1/8}{64d^2} & \text{if } x_d \in [0, 1/8d] \end{cases} \\
&< 1.
\end{aligned}$$

The volume of the ellipsoid is

$$\begin{aligned}
\operatorname{vol} E = \frac{\operatorname{vol} B(0,1)}{\sqrt{ab^{d-1}}} &= \frac{\operatorname{vol} B(0,1)}{\sqrt{(1 + 1/8d)(1 - 1/16d^2)^{d-1}}} \\
&\le \frac{\operatorname{vol} B(0,1)}{\sqrt{(1 + 1/8d)(1 - 1/16d)}} \le \operatorname{vol} B(0,1)(1 - 1/20d)
\end{aligned}$$

since $(1 + 1/8d)(1 - 1/16d)(1 - 1/40d)^2 \ge 1$ for all $d \ge 1$. □

Therefore, each $\Omega(d)$ iterations of the algorithm, we halve the volume of B'_{out}. Since in realistic computers a number of order 2^n needs at least n bits to encode, the number of iterations is bounded by a polynomial $O(dn)$ which

polynomial in d and size of the input. To make the algorithm realistic, we would need to mitigate the effects of rounding, and estimate the amount of memory to keep track of the current system of coordinates. We refer to the book by Grötschel, Lovász, Schrijver for the these (messy) details.

We note that on the line 12 the algorithm EllipsoidMethod uses the smallest ellipsoid rather than the ellipsoid E featured in the proof of the lemma above. Clearly, the smallest ellipsoid has smaller volume than E. It is not immediately clear whether the smallest ellipsoid can be computed quickly. It can be, and it is not overly laborious exercise to find its equation. The reader who wants to avoid it can substitute E on line 12.

Computing volumes: multistage probabilistic algorithm

By an application of ellipsoid method, and changing coordinate system, we can assume that the convex body is sandwiched between two balls $B(0,1)$ and $B(0,8d^2)$. The factor $8d^2$ between ball radii is called *sandwiching ratio*. With Darts algorithm this gives an algorithm of running time of $\exp(\Theta(d^2))$ to computer $\operatorname{vol} C$. That is too slow, but ellipsoid method is not to be blamed because even if C is a simplex, then the sandwiching ratio is d, and if C is a cube, it is $\sqrt{d}$ even for the best sandwiching.

The way over this obstacle is to have a nested sequence of "targets" on which we will use Darts algorithm. Suppose we have a sequence of convex bodies $B(0,1) = C_0 \subset C_1 \subset \cdots \subset C_n = C$, where $n = d^{O(1)}$ and we are able to pick a point uniformly at random from any C_i. Then we use Darts to compute each of the ratios

$$\frac{\operatorname{vol} C_0}{\operatorname{vol} C_1}, \frac{\operatorname{vol} C_1}{\operatorname{vol} C_2}, \ldots, \frac{\operatorname{vol} C_{n-1}}{\operatorname{vol} C_n}$$

with relative error at most $\delta/2n$ and error probability at most $1/10n$. Then with error probability at most $n \cdot 1/10n = 1/10$ we obtain an approximation of

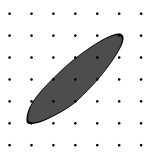
$$\frac{\operatorname{vol} B(0,1)}{\operatorname{vol} C} = \frac{\operatorname{vol} C_0}{\operatorname{vol} C_1} \cdot \cdots \cdot \frac{\operatorname{vol} C_{n-1}}{\operatorname{vol} C_n}$$

with relative error at most $(1+\delta/2n)^n \leq \exp(\delta/2) \leq 1+\delta$. If the ratios $\operatorname{vol} C_{i+1}/\operatorname{vol} C_i$ are all small, then the obtained algorithm will be fast.

It is easy to choose the sequence $C_0 \subset \cdots \subset C_n$ satisfying these conditions. We can set $C_i = B(0, 2^{i/d}) \cap C$. Then C_i's are convex and $\operatorname{vol} C_{i+1}/\operatorname{vol} C_i \leq 2$. The difficult part is to generate a random point inside C_i. If we have an oracle for C, then we can make an oracle for C_i by adding a "secretary" to our oracle, which checks whether the point is in $B(0, 2^{i/d})$ before passing the question to the oracle.

The surprising fact is that it *is possible* to efficiently sample a random point from an arbitrary convex body using only a membership oracle. The idea is to place a sufficiently fine grid inside C, and perform a random walk on the grid. We start with a known point inside C, and at each step the we move to one of

the adjacent grid points. Of course, if the step we are about to take will take us outside C, we will not take it. Since it is intuitively clear that the convex body does not have any "bottlenecks", the random walk is not confined to any small part of C, and after a while we should find at a random point of C. The basic problem with this approach is that it behaves poorly near the boundary of C. At the extreme, the set of grid points inside C might fail to form a connected graph under the adjacency relation.



Troublesome body

There are several ways to alleviate the boundary problems, and all of them involve smoothing C to remove the sharp corners. In the smoothing we will use, we replace C by a non-negative function f which is equal to 1 on C, and decays rapidly away from C. The function f will be a log-concave function (definition will be given below), and the integral of f will be at most twice larger than $\operatorname{vol} C$. We will then sample from the probability distribution with density proportional to f, and reject the sample if the point falls outside C. Since $\operatorname{vol} C / \int f \geq 1/2$, it will take us on average two samples to generate a uniformly distributed point in C.

Computing volumes: rapid mixing of Markov chains

We are about to employ a random walk to sample points from some log-concave probability distribution π. We will need to show after a while the position of the random walk will be distributed according to π, and that it will not take long. In other words, we wish to show that the random walk mixes fast. In this section we develop the probabilistic tool for the task.

A *Markov chain* on the state space Ω with initial is a sequence of random variables $X_0, X_1, \ldots$ taking values in Ω such that

$$\mathbf{P}[X_{t+1} = y | X_t = x, X_{t-1}, \ldots, X_0] = P(x, y).$$

One thinks of t as discrete time, with Markov chain starting in a state X_0, and at step making a transition to the next state according to a probability law that depends only on the current state. The simplest example is a random walk on a graph G: the state space is the vertex set $V(G)$, and at each step the chain picks uniformly at random an edge emanating from $v \in V$, and moves to the other endpoint of the edge.

A *stationary distribution* of a Markov chain is a function $\pi \colon \Omega \to [0, 1]$, which is a probability distribution (i.e. non-negative, and $\sum_x \pi(x) = 1$), such that

$$\pi(y) = \sum_{x \in \Omega} \pi(x) P(x, y) \qquad \text{for all } y \in \Omega.$$

In other words, a stationary distribution is the probability distribution such that if X_i is distributed according to π, then so are $X_{i+1}, X_{i+2}, \ldots$. For example, if G is a d-regular graph, then the uniform distribution $\pi(x) = 1/|V(G)|$ is a stationary distribution.

We will concern ourselves exclusively with *reversible* Markov chains. These are the chains for which there is a probability distribution $\pi\colon \Omega \to [0,1]$ such that

$$\pi(x)P(x,y) = \pi(y)P(y,x) \qquad \text{for all } x,y \in \Omega. \tag{1}$$

Such a π is a stationary distribution because $\sum_x \pi(x)P(x,y) = \sum_x \pi(y)P(y,x) = \pi(y)\sum_x P(y,x) = \pi(y)$. The condition is called reversibility because (1) says that when reversible Markov chain is in a stationary distribution, the transition from x to y is as likely as the transition from y to x. The other way to write reversibility is

$$\hat{P}(x,y) = \hat{P}(y,x),$$

where $\hat{P}(x,y) = P(x,y)/\pi(y)$.

The canonical example of a reversible Markov is a random walk on a graph G. In that case $\pi(v) = 1/\deg v$ is a stationary distribution satisfying (1). The case of a general reversible Markov chain is only little more general than this special case. It corresponds to a random walk on a weighted graph, in which the edge uv gets the weight $\pi(u)P(u,v)$, and at each step the edge is selected with a probability proportional to its weight. The reader, who when reading the foregoing exposition, thinks of the special case of a random walk on an unweighted graph will lose no essential idea.

Under very mild condition on a Markov chain, the stationary distribution is unique. The two conditions are irreducibility (one can reach from each state to every other), and aperiodicity (the number of steps to go from a state to itself is not always a multiple of the same integer). We will not show the uniqueness here. For our case it will be a part of a stronger conclusion that follows from the stronger assumptions available at our disposal. One of these assumptions will be that the Markov chain is *lazy*, that is to say it satisfies $P(x,x) = 1/2$ for all $x \in \Omega$. A lazy chain is clearly aperiodic. The irreducibility will follow from a bound on conductance which will now define.

For a Markov chain on a finite state space Ω the *capacity* of $S \subset \Omega$ is

$$\sum_{x \in S} \pi(x),$$

i.e. the probability that a Markov chain in a stationary state is in S, the *flow* out of $S \subset \Omega$ is

$$F_S \stackrel{\text{def}}{=} \sum_{\substack{x \in S \\ y \notin S}} \pi(x)P(x,y),$$

which is the probability that a Markov chain in a stationary state leaves S. The *conductance* of a non-empty set $S \subset \Omega$ is the ratio $\Phi_S \stackrel{\text{def}}{=} F_S/C_S$. It is the conditional probability of leaving S starting from S. Finally, the conductance of the Markov chain is

$$\Phi \stackrel{\text{def}}{=} \min_{\substack{S \subset \Omega \\ C_S \le 1/2}} \Phi_S.$$

The conductance is small if there is a "bottleneck", i.e. a set S of states that is hard to escape. If the Markov chain starts distributed according to π, then

it takes time $1/\Phi_S$ just to escape S. Thus we cannot expect the random walk to mix fast if Φ is small. The following result shows that the converse is true as well.

We hope that $\mathbf{P}[X_t = x]$ approaches $\pi(x)$ as $t \to \infty$. Let

$$\Delta_t(x) = \mathbf{P}[X_t = x] - \pi(x),$$

be the deviation between our hope and the reality at time t. We shall mostly work with its relative version

$$\Delta'_t(x) = \Delta_t(x)/\pi(x).$$

Define

$$d_2(t) = \sum_{x\in\Omega} \Delta_t(x)^2/\pi(x) = \sum_{x\in\Omega} \pi(x)\Delta'_t(x)^2.$$

This is often called the chi-squared distance to the stationary distribution. The following theorem shows that if Φ is large, these distances decay quickly.

Theorem 7 (Jerrum–Sinclair)**.** *A lazy reversible Markov chain on a finite state space satisfies*

$$d_2(t+1) \le (1-\Phi^2)d_2(t).$$

In particular, $d_2(t) \le d_2(0)(1-\Phi^2)^t$.

The proof will be based on two lemmas.

Lemma 8. *For every reversible lazy Markov chain*

$$d_2(t+1) \le d_2(t) - \tfrac{1}{2}\sum_{x\in\Omega}\sum_{y\in\Omega}\big(\Delta'_t(y) - \Delta'_t(x)\big)^2\pi(x)P(x,y)$$

Proof. Define

$$P'(x,y) = \begin{cases} P(x,y) & \text{if } x \ne y, \\ P(x,x) - 1/2 & \text{if } x = y. \end{cases}$$

Note that

$$\begin{aligned}
d_2(t) = \sum_{x\in\Omega} \Delta'_{t+1}(x)^2\pi(x) &= \sum_{x\in\Omega}\sum_{y\in\Omega} \pi(x)\Delta'_t(x)^2P(x,y) && \text{as } \sum_y P(x,y) = 1 \\
&= \sum_{x\in\Omega}\sum_{y\in\Omega} \pi(y)\Delta'_t(x)^2P(y,x) && \text{reversibility} \\
&= \sum_{x\in\Omega}\sum_{y\in\Omega} \pi(y)(\Delta'_t(x)^2 + \Delta'_t(y)^2)P'(y,x) && \text{laziness} \\
&= \sum_{x\in\Omega} \pi(x)\sum_{y\in\Omega}\big(\Delta'_t(x)^2 + \Delta'_t(y)^2\big)P'(x,y) && \text{renaming.}
\end{aligned}$$

To relate this quantity to $d_2(t+1)$ we first note that

$$\begin{aligned}
\Delta_{t+1}(x) = \mathbf{P}[X_{t+1} = x] - \pi(x) &= \sum_{y\in\Omega} \big(\mathbf{P}[X_t = y] - \pi(y)\big) P(y,x) \\
&= \sum_{y\in\Omega} \Delta_t(y) P(y,x) \\
&= \sum_{y\in\Omega} \Delta'_t(y) \pi(x) P(x,y) \\
&= \sum_{y\in\Omega} \big(\Delta'_t(y) + \Delta'_t(x)\big) \pi(x) P'(x,y) && \text{laziness}
\end{aligned}$$

This allows us to massage the expression for $d_2(t+1)$ into a suitable form by means of a Cauchy–Schwarz inequality

$$\begin{aligned}
d_2(t+1) = \sum_{x\in\Omega} \Delta_{t+1}(x)^2/\pi(x) &= \sum_{x\in\Omega} \pi(x) \left(\sum_{y\in\Omega} \big(\Delta'_t(y) + \Delta'_t(x)\big) P'(x,y) \right)^2 \\
&\le \sum_{x\in\Omega} \pi(x) \left(\sum_{y\in\Omega} \big(\Delta'_t(y) + \Delta'_t(x)\big)^2 P'(x,y) \right) \left(\sum_{\substack{y\in\Omega \\ y\neq x}} P'(x,y) \right) && \text{Cauchy–Schwarz} \\
&= \tfrac{1}{2} \sum_{x\in\Omega} \pi(x) \left(\sum_{y\in\Omega} 2\big(\Delta'_t(y)^2 + \Delta'_t(x)^2\big) - \big(\Delta'_t(y) - \Delta'_t(x)\big)^2 P'(x,y) \right) && \text{laziness} \\
&= d_2(t) - \tfrac{1}{2} \sum_{x\in\Omega} \pi(x) \sum_{y\in\Omega} \big(\Delta'_t(y) - \Delta'_t(x)\big)^2 P'(x,y) \\
&= d_2(t) - \tfrac{1}{2} \sum_{x\in\Omega} \pi(x) \sum_{y\in\Omega} \big(\Delta'_t(y) - \Delta'_t(x)\big)^2 P(x,y). \quad \square
\end{aligned}$$

Recall the notation $[d] \stackrel{\text{def}}{=} \{1, 2, \ldots, d\}$.

Lemma 9. *Suppose $\Omega = [n]$ and $f\colon \Omega \to \mathbb{R}$ satisfies $f(1) \ge f(2) \ge \cdots \ge f(d)$. Assume furthermore that there is $w \in \Omega$ such that $f(w+1) = 0$ and the capacity of $[w]$ is $C_{[w]} \le 1/2$, then*

$$\sum_{x<y} \pi(x) P(x,y) \big(f(x) - f(y)\big)^2 \ge \Phi^2 \sum_{x\le w} f(x)^2 \pi(x)$$

Proof. Consider

$$\begin{aligned}
\sum_{x<y} \pi(x) P(x,y) \big(f(x)^2 - f(y)^2\big) &= \sum_{x<y} \pi(x) P(x,y) \sum_{x\le z<y} \big(f(z)^2 - f(z+1)^2\big) \\
&= \sum_{z} \big(f(z)^2 - f(z+1)^2\big) \sum_{x\in S_z} \pi(x) P(x,y) \\
&= \sum_{z} \big(f(z)^2 - f(z+1)^2\big) F_{[z]}
\end{aligned}$$

which since $C_{S_z} \le C_S \le 1/2$ for $z \le w$ and $f(z) \ge f(z+1)$ can be bounded by

$$\begin{aligned}
&\ge \Phi \sum_{z \le w} \big(f(z)^2 - f(z+1)^2\big) C_{[z]} \\
&= \Phi \sum_{z \le w} \big(f(z)^2 - f(z+1)^2\big) \sum_{x \le z} \pi(x) \\
&= \Phi \sum_{x \le w} \pi(x) \sum_{x \le z \le w} \big(f(z)^2 - f(z+1)^2\big) \\
&= \Phi \sum_{x \le w} \pi(x) \big(f(x)^2 - f(w+1)^2\big) \\
&= \Phi \sum_{x \le w} \pi(x) f(x)^2
\end{aligned}$$

Therefore, we need to find a suitable upper bound on $\sum_{x<y} \pi(x)P(x,y)(f(x)^2 - f(y)^2)$. It is done by a clever use of the Cauchy–Schwarz inequality

$$\begin{aligned}
\left(\sum_{x<y} \pi(x)P(x,y)\big(f(x)^2 - f(y)^2\big)\right)^2 &= \left(\sum_{x<y} \pi(x)P(x,y)\big(f(x) - f(y)\big)\big(f(x) + f(y)\big)\right)^2 \\
&\le \sum_{x<y} \pi(x)P(x,y)\big(f(x) - f(y)\big)^2 \\
&\qquad \times \sum_{x<y} \pi(x)P(x,y)\big(f(x) + f(y)\big)^2
\end{aligned}$$

The first term of the product is exactly what we are after. The second term can be estimated by

$$\begin{aligned}
\sum_{x<y} \pi(x)P(x,y)(f(x)+f(y))^2 &\le 2 \sum_{x<y} \pi(x)P(x,y)(f(x)^2 + f(y)^2) \\
&= 2 \sum_x f(x)^2 \left(\sum_{x<y\le w} \pi(x)P(x,y) + \sum_{y<x} \pi(y)P(y,x) \right) \\
&= 2 \sum_x f(x)^2 \sum_{\substack{y \ne x \\ y \le w}} \pi(x)P(x,y) && \text{reversibility} \\
&\le \sum_x \pi(x) f(x)^2 && \text{laziness}
\end{aligned}$$

Putting everything together we obtain

$$\left(\Phi \sum_{x \le w} \pi(x) f(x)^2\right)^2 \le \left(\sum_{x \le w} \pi(x) f(x)^2\right) \left(\sum_{x<y} \pi(x)P(x,y)\big(f(x) - f(y)\big)^2\right)$$

from which it follows that

$$\sum_{x<y} \pi(x)P(x,y)\big(f(x) - f(y)\big)^2 \ge \Phi^2 \sum_{x \le w} f(x)^2 \pi(x). \qquad \square$$

Finally we can give a proof of Theorem 7.

Proof. Order Ω so that $\Delta'(1) \geq \cdots \geq \Delta'(n)$. Let w be the largest integer such that $C_{[w]} \leq 1/2$. Set

$$f_1(x) = \max(\Delta'(x) - \Delta'(w+1), 0)$$
$$f_2(x) = \max(\Delta'(w+1) - \Delta'(x), 0).$$

The functions f_1 and f_2 are the positive and negative parts of $\Delta'(x) - \Delta'(w+1)$. Clearly $(\Delta'(x) - \Delta'(y))^2 \geq (f_1(x) - f_1(y))^2 + (f_2(x) - f_2(y))^2$. Applying the Lemma 9 to f_1 and f_2 in turn we obtain

$$\sum_{x<y} \pi(x)P(x,y)\big(f_1(x) - f_1(y)\big)^2 \geq \Phi^2 \sum_x f_1(x)^2\pi(x)$$
$$\sum_{x<y} \pi(x)P(x,y)\big(f_2(x) - f_2(y)\big)^2 \geq \Phi^2 \sum_x f_2(x)^2\pi(x)$$

Adding these two formulas together we obtain

$$\begin{aligned}\sum_{x<y} \pi(x)P(x,y)\big(\Delta'(x) - \Delta'(y)\big)^2 &\geq \Phi^2 \sum_{x\leq w} (f_1(x)^2 + f_2(x)^2)\pi(x) \\ &= \Phi^2 \sum_x \big(\Delta'(x) - \Delta'(w+1)\big)^2\pi(x)\end{aligned}$$

which, since $\sum_x \Delta'(x)\pi(x) = \sum_x \Delta(x) = 0$, is equal to

$$\begin{aligned}&= \Phi^2 \sum_x \big(\Delta'(x)^2 + \Delta'(w+1)^2\big)\pi(x) \\ &\geq \Phi^2 \sum_x \Delta'(x)^2.\end{aligned}$$

Plugging this into Lemma 8 the theorem follows. □

Problems

1. Let P be a partially ordered set on the ground set $[n] = \{1, 2, \ldots, n\}$. Let C be a convex polytope given by the inequalities

   $$\begin{aligned} x_i &\geq 0 && \text{for } i = 1, \ldots, n, \\ x_i &\leq x_j && \text{if } i \prec_P j. \end{aligned}$$

   Show that $\mathrm{vol}(P)$ is proportional to the number of linear extensions of P. What is the proportionality constant?

2. (a) Suppose $P_0 \subset \mathbb{R}^d$ is arbitrary. Define sets $P_1, P_2, \ldots$ inductively by $P_{i+1} = \{\lambda p + (1-\lambda)q : p, q \in P_i,\ 0 \leq \lambda \leq 1\}$. Show that $\mathrm{conv}(P) = \bigcup_i P_i$.

(b) Show that $\text{conv}(P) = P_d$.

3. Given an oracle that generates random uniformly distributed real numbers from the interval $[0, 1]$, find a way to generate a uniformly distributed point in $B(0, 1) \subset \mathbb{R}^d$ by asking only d oracle queries. (Generate some other radially symmetric distribution.)

4. Show that algorithm DARTS works even if the points generated at different times at line 6 are not independent from one another, but only pairwise independent.

5. Suppose C is a positive real number, and X is a random variable such $\mathbf{P}[|X - Y| \leq \delta Y] < 1/10$. Suppose $X_1, \ldots, X_n$ are independent and are distributed identically with the same distribution as X. Let X_m be the median of $X_1, \ldots, X_n$. Show that $\mathbf{P}[|X_m - Y| \leq \delta Y] < 2^{-n}$.

6. Show that if $C \subset \mathbb{R}^d$ is convex, then $\text{vol}\big(C \cap B(0, r)\big) \leq r^d \,\text{vol}(C \cap B(0, 1))$.

7. Modify the ellipsoid method as to reduce the ratio between the radii of B'_{in} and B'_{out} from $8d^2$ to $8d\sqrt{d}$. [Hint: use different points at distance $1/8d$ at the line 6 of ELLIPSOIDMETHOD].