Solutions to Homework 3

Probability Background

In this section, we review some mathematical concepts which will be used throughout the rest of homework set. This part contains no problems but it is recommended, since you will be responsible to understand these ideas. We begin with some basic definitions and facts. By a *random variable*, we mean a function $X : \Omega \to \mathbb{R}$ where we think of Ω as the set of possible outcomes of some experiment, and $X(\omega)$ the measure of the outcome $\omega \in \Omega$. We say X is a $0/1-$random variable (resp. *integer random variable*, *non-negative random variable*) if $X(\omega) \in \{0, 1\}$ (resp. $X(\omega) \in \mathbb{Z}$, $X(\omega) \geq 0$) for all $\omega \in \Omega$. We generally hide Ω and instead, identify X with its image or *support* in $\mathbb{R}$, denoted $\mathrm{Supp}(X) \subset \mathbb{R}$. For $x \in \mathrm{Supp}(X)$, we let $p_X(x) = \Pr_{\omega \in \Omega}\big[X(\omega) = x\big]$. When the random variable X is clear from context, we write $p(x)$ instead of $p_X(x)$.

Mathematical Aside: For us, $\big|\mathrm{Supp}(X)\big|$ will always be at most countably infinite (this allows us to sum over all $x \in \mathrm{Supp}$ rather than having to integrate); a typical, commonly occuring case is when $\mathrm{Supp}(X) = \mathbb{Z}$.

Definition (Expectation of a Random Variable). Let X be a random variable. The *expectation* of X, denoted $\mathbb{E}[X]$ is

$$\mathbb{E}[X] := \sum_{x \in \mathrm{Supp}(X)} p(x) \cdot x.$$

Fact (Linearity of Expectation). The expectation operator is linear. For real random variables X and Y with the same support and $\alpha, \beta \in \mathbb{R}$:

$$\begin{aligned}
\mathbb{E}[\alpha X + \beta Y] &= \sum_{x,y \in \mathrm{Supp}(X)} \Pr\big[X = x \ \&\ Y = y\big] \cdot (\alpha x + \beta y) \\
&= \sum_{x,y \in \mathrm{Supp}(X)} \alpha x \cdot \Pr\big[X = x \ \&\ Y = y\big] + \sum_{x,y \in \mathrm{Supp}(X)} \beta y \cdot \Pr\big[X = x \ \&\ Y = y\big] \\
&= \sum_{x \in \mathrm{Supp}(X)} \alpha x \sum_{y \in \mathrm{Supp}(Y)} \Pr\big[X = x \ \&\ Y = y\big] + \\
&+ \sum_{y \in \mathrm{Supp}(X)} \beta y \sum_{x \in \mathrm{Supp}(X)} \Pr\big[X = x \ \&\ Y = y\big] \\
&= \alpha \cdot \sum_{x \in \mathrm{Supp}(X)} p_X(x) x + \beta \cdot \sum_{y \in \mathrm{Supp}(X)} p_Y(y) y = \alpha \mathbb{E}[X] + \beta \mathbb{E}[Y].
\end{aligned}$$

Fact (Markov's Inequality). Let X be a non-negative random variable (*i.e.*, $x \geq 0$ for all $x \in \mathrm{Supp}(X)$). Then for all $a \in \mathbb{R}$,

$$\Pr\big[X \geq a\big] \leq \frac{\mathbb{E}[X]}{a}.$$

Proof. We have

$$\mathbb{E}[X] = \sum_{x \in \mathrm{Supp}(X)} p(x) \cdot x = \sum_{x \geq a} p(x) \cdot x + \sum_{x < a} p(x) \cdot x \geq a \sum_{x \geq a} p(x) = a \cdot \Pr\big[X \geq a\big].$$

Markov's inequality follows (note X non-negative is needed to ensure $\sum_{x<a} p(x) \cdot x \geq 0$). □

The Chernoff-Hoeffding Inequality

The Chernoff-Hoeffding inequality is a very useful "tail bound" which bounds the probability that the mean of several independent copies of a random variable deviates from its expectation.

Theorem (The Chernoff-Hoeffding Inequality). Let $X_1, \ldots, X_k$ be independent, identically distributed (i.i.d.) copies of a 0/1 random variable and let $p = \mathbb{E}[X] = \Pr\big[X = 1\big]$. Then for all $\varepsilon > 0$ such that $3\varepsilon \leq p$,

$$\Pr\left[\left|\frac{X_1 + \cdots + X_k}{k} - p\right| > \varepsilon\right] \leq 2e^{-k\varepsilon^2/3p}. \tag{1}$$

Remark.

1. In many applications (for example, for the application required to do problem 6), p and ε are constants, in which case the right hand side of (1) is $e^{-\Omega(k)}$, which says, in words, that the probability the average of the X_i is far from its expectation decays *exponentially* with k.

2. Tighter and more general forms of the above theorem are known to hold. For example, versions hold where we don't insist that the X_i are 0/1 random variables, or where we only have $t-$wise independence of the X_i for $t < k$ instead of full independence. These versions are useful as well. However, in practice, it is good to know how to *derive* inequalities of Chernoff-Hoeffding type in case the precise form you need does not appear in literature.

Proof. Let $X = X_1 + \cdots + X_k$. We prove $\Pr\big[X > k(p+\varepsilon)\big] \leq e^{-k\varepsilon^2/3p}$; a similar argument can be used to show that $\Pr\big[X < k(p-\varepsilon)\big] \leq e^{-k\varepsilon^2/3p}$, from which (1) follows. For all $\alpha, t \in \mathbb{R}$ such that $t > 0$, we have

$$\begin{aligned}
\Pr\big[X > \alpha kp\big] &= \Pr\big[e^{tX} > e^{\alpha tkp}\big] \leq e^{-\alpha tkp} \cdot \mathbb{E}[e^{tX}] = e^{-\alpha tkp} \cdot \mathbb{E}\big[e^{t(X_1 + \cdots + X_k)}\big] \\
&= e^{-\alpha tkp} \cdot \mathbb{E}\Big[\prod_{i=1}^{k} e^{tX_i}\Big] = e^{-\alpha tkp} \cdot \prod_{i=1}^{k} \mathbb{E}[e^{tX_i}],
\end{aligned}$$

where the inequality on the first line is Markov's inequality, and the final equality used the independence of the X_i. Note that for all $i = 1, \ldots, k$,

$$\mathbb{E}[e^{tX_i}] = p \cdot e^t + (1-p) = p \cdot (e^t - 1) + 1 \leq e^{p(e^t-1)},$$

using the bound $1 + z \leq e^z$ which holds for all z. Plugging this in gives

$$\Pr\big[X > \alpha kp\big] \leq e^{-kp(\alpha t - e^t + 1)} \leq e^{-kp(\ln(\alpha)\cdot\alpha - \alpha + 1)},$$

where the final inequality is obtained by setting $t = \ln(\alpha)$, the value where the function $f(t) = \alpha t - e^t + 1$ achieves its minimum. Finally, we set $\alpha = 1 + \varepsilon/p$ and get

$$\Pr\big[X > k(p+\varepsilon)\big] \leq e^{-kp(\ln(1+\varepsilon/p)\cdot(1+\varepsilon/p) - \varepsilon/p)},$$

and so it remains to show that $\ln(1+\varepsilon/p)\cdot(p+\varepsilon) \geq \varepsilon + \varepsilon^2/3p$. This follows from the bound $\ln(1+z) \geq z - z^2/2$, which holds for all $|z| < 1$ (follows from the Taylor series expansion of the function $f(z) = \ln(1+z) = \sum_{n\geq 1}(-1)^{n+1}z^n/n$). We get

$$\ln(1+\varepsilon/p)\cdot(p+\varepsilon) \geq (p+\varepsilon)\cdot\big(\varepsilon/p - \varepsilon^2/2p^2\big) = \varepsilon + \varepsilon^2/2p - \varepsilon^3/2p^2 \geq \varepsilon + \varepsilon^2/3p,$$

where the final inequality has used $3\varepsilon \leq p$. □

Robustness of BPP

In this section we consider two changes to the definition of decision of a probabilistic Turing Machine from class. We show that neither change affects the definition of BPP.

Problem 1. Let $f : \{0,1\}^* \to \{0,1\}$ be a function. Suppose there exists a polynomial time PTM M such that for all $\mathbf{x} \in \{0,1\}^*$ of size $|\mathbf{x}| = n$:

$$\Pr_{\mathbf{r}}\big[\mathsf{M}(\mathbf{x},\mathbf{r}) = f(\mathbf{x})\big] \geq \frac{1}{2} + n^{-c},$$

for a constant c. Prove that for any other constant d, there is another polynomial time PTM M' such that for all $\mathbf{x} \in \{0,1\}^*$ of size $|\mathbf{x}| = n$:

$$\Pr_{\mathbf{r}}\big[\mathsf{M}'(\mathbf{x},\mathbf{r}) = f(\mathbf{x})\big] \geq 1 - 2^{-n^d}.$$

(**Hint:** Use the Chernoff-Hoeffding inequality).

Solution. Given M we define another probabilistic Turing Machine M' which, on input $\mathbf{x} \in \{0,1\}^n$, runs M $k = n^{2c+d+1}$ times, obtaining outputs $b_1, \ldots, b_k \in \{0,1\}$. M' outputs 1 if $b_i = 1$ for at least half of the $i \in \{1, \ldots, k\}$, and outputs 0 otherwise. We analyze the success probability of M' (*i.e.*, we bound $\Pr\big[\mathsf{M}'(\mathbf{x}) = f(\mathbf{x})\big]$). Note the b_i are independent 0/1 random variables with the same expectation; this expectation is $\leq 1/2 - n^{-c}$ if $f(\mathbf{x}) = 0$ and is $\geq 1/2 + n^{-c}$ if $f(\mathbf{x}) = 1$. Let $b = b_1 + \cdots + b_k$. If $f(\mathbf{x}) = 0$ then the expectation of b is $\leq \frac{k}{2}(1 - n^{-c})$, while if $f(\mathbf{x}) = 1$, the expectation of b is $\geq \frac{k}{2}(1 + n^{-c})$. Note $\mathsf{M}'(\mathbf{x})$ outputs 1 if and only if $b \geq \frac{k}{2}$, and so $\mathsf{M}'(\mathbf{x}) \neq f(\mathbf{x})$ only occurs if $\mathbb{E}[b] \leq \frac{k}{2}(1 - n^{-c})$ and $b \geq \frac{k}{2}$ or if $\mathbb{E}[b] \geq \frac{k}{2}(1 + n^{-c})$ but $b < \frac{k}{2}$. In any case, we set $\varepsilon = \frac{n^{-c}}{4}$ and get

$$\Pr\big[\mathsf{M}'(\mathbf{x}) \neq \mathsf{L}(\mathbf{x})\big] \leq \Pr\left[\left|\frac{b_1 + \cdots + b_k}{k} - \mathbb{E}[b_i]\right| > \varepsilon\right] \leq 2e^{-k\varepsilon^2/3p} \leq 2^{-n^d},$$

using the Chernoff-Hoeffding bound.

Problem 2. Suppose $f : \{0,1\}^* \to \{0,1\}$ is computed by a PTM M which runs in *expected* polynomial time (*i.e.*, the random variable depicting M's runtime on inputs of size n has expectation which is $\mathcal{O}(n^c)$ for some constant c). Prove that f can be computed by another PTM M' which runs in strict polynomial time.

Solution. Let M be a PTM which computes f in expected polynomial time. Let $\mathsf{T}(n)$ be the expected running time of M on inputs of size n (so $\mathsf{T}(n) = n^c$ for some constant c). Consider the PTM M' which runs M for $100 \cdot \mathsf{T}(n)$ steps and then halts with output 0 if M has not already halted. Note M' runs in strict polynomial time. Moreover, Markov's inequality gives for all $\mathbf{x} \in \{0,1\}^*$,

$$\Pr\big[\mathsf{M}'(\mathbf{x}) \neq \mathsf{M}(\mathbf{x})\big] \leq \Pr\big[\mathsf{M} \text{ runs for } \leq 100 \cdot \mathsf{T} \text{ steps}\big] \leq \frac{\mathsf{T}}{100 \cdot \mathsf{T}} = \frac{1}{100}.$$

It follows that

$$\Pr\big[\mathsf{M}'(\mathbf{x}) = f(\mathbf{x})\big] \geq \Pr\big[\mathsf{M}(\mathbf{x}) = f(\mathbf{x})\big] - \Pr\big[\mathsf{M}'(\mathbf{x}) \neq \mathsf{M}(\mathbf{x})\big] \geq \frac{2}{3} - \frac{1}{100} \geq .65,$$

and so M' decides L in strict polynomial time.

Statistical Distance

Statistical distance is a basic mathematical notion which captures the distance between two probability distributions.

Definition (Statistical Distance). Let X and Y be two probability distributions with the same support. The *statistical distance* between X and Y, denoted $\Delta(X,Y)$ is

$$\Delta(X,Y) := \frac{1}{2} \sum_{x \in \mathrm{Supp}(X)} \Big| p_X(x) - p_Y(x) \Big|.$$

Problem 3. Suppose the PTM M computes $f : \{0,1\}^* \to \{0,1\}$ using m bits of randomness (*i.e.*, during the first step of computation M writes a random string of length m onto its random tape). Let X be a probability distribution on $\{0,1\}^m$ such that $\Delta(X, U_m) \leq 1/10$, where U_m denotes the uniform distribution on $\{0,1\}^m$. Show that M can be used to compute f even if X is used to populate M's random tape, instead of U_m.

Solution. For $\mathbf{x} \in \{0,1\}^*$, let $\mathsf{G} \subset \{0,1\}^m$ be the set of *good* random strings $\mathbf{r} \in \{0,1\}^m$ which make $\mathsf{M}(\mathbf{x}, \mathbf{r}) = f(\mathbf{x})$. Note $\Pr_{\mathbf{r} \sim \{0,1\}^m}\big[\mathbf{r} \in \mathsf{G}\big] \geq 2/3$ since the PTM M computes f. Also, note that if M's random tape is populated by $\mathbf{r} \sim X$, then $\mathsf{M}(\mathbf{x}, \mathbf{r}) = f(\mathbf{x})$ occurs with probability $\Pr_{\mathbf{r} \sim X}\big[\mathbf{r} \in \mathsf{G}\big]$. We have

$$\Pr_{\mathbf{r} \sim X}\big[\mathbf{r} \in \mathsf{G}\big] = \sum_{\mathbf{r} \in \mathsf{G}} p_X(\mathbf{r}) = \sum_{\mathbf{r} \in \mathsf{G}} 2^{-m} + \sum_{\mathbf{r} \in \mathsf{G}} \big(p_X(\mathbf{r}) - 2^{-m}\big) \geq \frac{2}{3} - |A|,$$

where $A := \sum_{\mathbf{r}\in\mathsf{G}}\big(p_X(\mathbf{r}) - 2^{-m}\big)$. Let $B := \sum_{\mathbf{r}\notin\mathsf{G}}\big(p_X(\mathbf{r}) - 2^{-m}\big)$. Note $A + B = 0$ and so $|A| = |B|$. Thus,

$$2|T| = |S| + |T| \leq \sum_{\mathbf{r}\in\{0,1\}^m} \big|p_X(\mathbf{r}) - 2^{-m}\big| = 2\Delta(X, U_m).$$

So we get $\Pr_{\mathbf{r}\sim X}\big[\mathbf{r} \in \mathsf{G}\big] \geq 2/3 - \Delta(X, U_m) \geq 2/3 - .1 > .55$, and so M still decides f even when its random tape is populated with strings drawn from X.

Simulating a Fair Coin with an Unfair Coin

Problem 4. Let $p \in \mathbb{R}$ be a real number such that $0 < p < 1$. Let BPP_p denote the set of functions $f : \{0,1\}^* \to \{0,1\}$ which can be computed by polynomial time PTMs whose random tapes are populated with bits drawn according to the Bernoulli distribution with parameter p (*i.e.*, each bit is 1 with probability p, 0 w.p. $1-p$). Prove that $\mathsf{BPP}_p = \mathsf{BPP}$.

Solution. We describe a procedure which outputs a random sample in $\{0,1\}^m$ given a randomness tape populated according to the Bernoulli distribution with parameter p. The procedure iterates over the Bernoulli tape, each time it reads a pair of adjacent bits (b, b'); if $(b, b') = (0, 1)$ it writes a 0 on the output tape; if $(b, b') = (1, 0)$ it writes a 1; otherwise it continues to the next pair. Note that the chance of writing a bit to the output tape is $2p(1-p)$ and so the expected number of Bernoulli pairs needed to generate m bits of output is $m/2p(1-p)$. Finally, note that the chance of writing a 0 is $p(1-p)$ which is the same as the chance of writing a 1. Therefore, the output distribution is uniform.

It follows that $\mathsf{BPP} \subset \mathsf{BPP}_p$ since any PTM who can decide a language with access to a uniform random string can be used to decide the same language given access to a Bernoulli string: first use the Bernoulli string to generate a uniform string, then use the original PTM.

Randomized Primality Testing

In this section we describe the Miller-Rabin randomized algorithm for primality testing. There are no questions in this section and this algorithm requires a heavy amount of algebra so feel free to skip this part. However, if you are curious then feel free to read and enjoy! As mentioned in class, a deterministic prime test algorithm, due to Argawal, Kayal and Saxena, is now known.

Recall that an integer N is *prime* if it cannot be written as a non-trivial product of two integers $N = a \cdot b$ (*i.e.* if $N = ab$ for integers a, b then either $a = 1, b = N$ or vice versa). Our randomized algorithm will efficiently compute the function f_{isPrime} which, on input N (represented in binary) outputs 1 if N is prime, 0 if N is composite. Notice that the size of N is the number of bits in its binary representation (roughly $\log N$), and so in this case "polynomial time" means time $\log^c N$ for a constant c.

The algorithm which computes f_{isPrime} works as follows:

Input: An integer N.

0. If $N = 2$, output 1; if N is any other even number, output 0; from now on assume N odd.

1. Choose $a \leftarrow \{1, \ldots, N-1\}$ at random; if $\mathsf{gcd}(a, N) > 1$, output 0 and halt.

2. Otherwise, let $N-1 = 2^r s$ for an odd integer s. Let $x_i = a^{2^i s} \pmod N$ for $i = 0, 1, \ldots, r-1$. If $x_0 \equiv 1 \pmod N$ or $x_i \equiv -1 \pmod N$ for $i = 1, \ldots, r-1$, output 1, otherwise output 0.

The core of the analysis of the algorithm consists of the following two claims. With these claims it is easy to show that $f_{\mathsf{isPrime}} \in \mathsf{BPP}$ using an amplification procedure as seen in class.

Claim 1. If N is an odd prime, then for all $a \in \{1, \ldots, N-1\}$, either $x_0 \equiv 1 \pmod N$ or else $x_i \equiv -1 \pmod N$ for some $i = 1, \ldots, r-1$. In other words, when $f_{\mathsf{isPrime}}(N) = 1$, the algorithm always outputs 1.

Claim 2. If N is composite, then the algorithm outputs 0 with probability at least $1/2$.

Proof of Claim 1. This proof uses without proof two facts from elementary number theory.

1. If N is prime, then for all $a \in \{1, \ldots, N-1\}$, $a^{N-1} \equiv 1 \pmod N$; this is called Fermat's Little Theorem.

2. If N is prime, then $X = \pm 1$ are the only solutions to the equation $X^2 \equiv 1 \pmod N$.

The first fact says that $x_r = a^{2^r s} = a^{N-1} \equiv 1 \pmod N$. Thus, $x_{r-1}^2 = x_r \equiv 1 \pmod N$, and so x_{r-1} is a solution to $X^2 \equiv 1 \pmod N$; by the second fact $x_{r-1} = \pm 1$. If $x_{r-1} \equiv -1$ we are done, and if $x_{r-1} \equiv 1 \pmod N$, then since $x_{r-2}^2 = x_{r-1}$, x_{r-2} is a solution to $X^2 \equiv 1 \pmod N$. Continuing in this fashion, we see either that some $x_i \equiv -1 \pmod N$ holds, or else $x_0 \equiv 1 \pmod N$ holds. In either case, the algorithm outputs 1. □

Proof of (a special case of) Claim 2. We prove Claim 2 in the special case when composite $N = p \cdot q$ for primes $p, q > 3$. The same methods can be extended to handle the general case. We will use without proof two more facts from elementary number theory.

1. If $N = p \cdot q$, then for all $(a, b) \in \{1, \ldots, p-1\} \times \{1, \ldots, q-1\}$, there exists $x \in \{1, \ldots, N-1\}$ such that $x \equiv a \pmod p$ and $x \equiv b \pmod q$; this is called the Chinese remainder theorem.

2. Suppose a set $S \subset \{1, \ldots, N-1\}$ is multiplicative (*i.e.*, if $x, y \in S$ then $x \cdot y \pmod N$ is also in S), and is proper. Then $|S| \leq (N-1)/2$.

Now, we prove that if $N = p \cdot q$ then the subset of a which cause the algorithm to output 1 is contained in some proper multiplicative subset. Therefore, by fact 2, the number of such a is at most $(N-1)/2$, and so the algorithm outputs 1 with probability at most $1/2$. Now, for $i = 1, \ldots, r-1$, let $S_i = \{a : a^{2^i s} = \pm 1 \pmod N\}$. Note S_i is multiplicative since if $a, b \in S_i$ then $(ab)^{2^i s} = a^{2^i s} \cdot b^{2^i s} = (\pm 1) \cdot (\pm 1) = \pm 1$. Now, let $i_0 \in \{1, \ldots, r-1\}$ be maximal such that $a_0^{2^{i_0} s} \equiv -1 \pmod N$ for some $a_0 \in \{1, \ldots, N-1\}$. Note that if $a \in \{1, \ldots, N-1\}$ makes the algorithm output 1, then either $a^s = 1$, or else there exists some $i \in \{1, \ldots, N-1\}$ such that $a^{2^i s} \equiv -1 \pmod N$. In the first case, $a \in S_i$ holds for all i, so in particular $a \in S_{i_0}$. In the second case, note that $a \in S_i$ for the i such that $a^{2^i s} \equiv -1 \pmod N$. Note $i \leq i_0$ since i_0 was maximal, and so $a \in S_{i_0}$ holds in the second case also. So we have shown that if a makes the algorithm output 1 then $a \in S_{i_0}$. It thus remains to show that S_{i_0} is proper. For this, we simply must show that some $x \notin S_{i_0}$. For this, we use the first fact. Let $x \in \{1, \ldots, N-1\}$ be such that $x \equiv a_0 \pmod p$ and $x \equiv 1 \pmod q$. Note $x^{2^{i_0} s} \equiv a_0^{2^{i_0} s} \equiv -1 \pmod p$, and $x^{2^{i_0} s} \equiv 1 \pmod q$. Thus, $x^{2^{i_0} s} \not\equiv \pm 1 \pmod N$, and so $x \notin S_{i_0}$, and we are done. □

Random Sample Consensus

In this section you will construct and analyze a simple case of the RanSaC algorithm.

Inputs. Your algorithm should take as input a set of N points $\{(x_i, y_i, z_i)\}_{i=1,\ldots,N}$ where each (x_i, y_i, z_i) is a triple of integers satisfying: $0 \leq x_i, y_i, z_i \leq 9999$.

Model Assumptions. Assume that the N triples you are given as input can be separated into two sets, GOOD and BAD such that:

- the good points all lie on a line $\ell \subset \mathbb{R}^3$;
- the bad points are distributed uniformly in $\{0, \ldots, 9999\}^3$.

Moreover, assume that at least 10% of your input points are in GOOD.

Problem 5. Describe a randomized algorithm that runs in $\mathcal{O}(\log N)$ time which, under the above model assumptions, recovers the line ℓ with 99% probability (the probability should be over your algorithm's randomness and the inputs).

Solution. Our algorithm makes use of a ScoreLine subroutine which takes as input a candidate line $\ell' \subset \mathbb{R}^3$ and outputs an approximate score of what fraction of the N points lie on ℓ'. It is parametrized by a constant k and works as follows:

- For $i = 1, \ldots, k$, draw a random input point (x_i, y_i, z_i) and check whether $(x_i, y_i, z_i) \in \ell'$.
- Let r denote the number of drawn points for which the check passes. Output r/k.

Notice that if $\ell' = \ell$ is the line we are looking for then we expect $r/k = .1$. By the Chernoff-Hoeffding bound, the probability that $\left|r/k - .1\right| > .03$ is at most $2e^{-.003k}$. We choose k large enough so that $2e^{-.003k} < .001 = .1\%$ ($k = 2700$ works). Thus, whenever $\ell' = \ell$, $r/k > .07$ holds with 99.9% probability. On the other hand, when $\ell' \neq \ell$ then we expect r/k to be very small. Specifically, note that ℓ' and ℓ intersect in at most one point, thus at most one good point chosen during ScoreLine can lie on ℓ'. Moreover, since the bad points are drawn randomly from $\{0, \ldots, 9999\}^3$, the probability that a bad point lies on ℓ' is roughly $\big(1/10000\big)^2$. Thus, again using the Chernoff-Hoeffding inequality, the chance that $r/k > .07$ when $\ell' \neq \ell$ is at most $.001/k'$ for some constant k' to be chosen later. Thus, the key takeaway from our subroutine is that when $\ell' = \ell$, the output is at least .07 with 99.9% probability, while when $\ell' \neq \ell$ the output is least .07 with $.001/k'$ probability. Note, finally that ScoreLine takes $\mathcal{O}(\log N)$ time.

Now we describe our main algorithm; this one is also parametrized by k', the constant mentioned above.

- For $i = 1, \ldots, k'$, draw two random input points $(x_i, y_i, z_i), (x_j, y_j, z_j)$ and let ℓ' be the line they span.
- Run ScoreLine(ℓ'), if the output is larger than .07 output ℓ, otherwise continue.

Note that the probability of choosing two good points is .01 and whenever this occurs, $\ell' = \ell$ (which means ScoreLine's output is greater than .07 with 99.9% probability). We choose k' large enough so that we expect two good points many times. Note the probability we never draw two good points is $(.99)^{k'}$ which is at most .1% when $k' = 1000$. Finally, the probability that we ever see a false positive is the probability that ScoreLine outputs a value larger than .07 when $\ell' \neq \ell$. By the above analysis this occurs with probability at most $(.001/k') * k' = .1\%$. It follows that our algorithm outputs ℓ correctly with probability 99%.

A Circuit Lower Bound

Recall the class $\mathsf{P}_{/\mathsf{POLY}}$ is the set of $f : \{0,1\}^* \to \{0,1\}$ which can be computed by a polynomial size family of circuits. It is open whether $\mathsf{NP} \subset \mathsf{P}_{/\mathsf{POLY}}$ or not; this is analogous to the P vs NP question. Showing that $\mathsf{NP} \not\subset \mathsf{P}_{/\mathsf{POLY}}$ ammounts to showing that the function f_{3SAT} (takes a 3CNF formula as input and outputs 1 if it is satisfiable, 0 if not) cannot be computed by a polynomial sized circuit. So far we have not had luck showing lower bounds on the circuit size required to compute most functions of interest. However, it is possible to show that *some* (in fact, most) functions must require large circuits.

Problem 6. Prove that there exists a function $f : \{0,1\}^n \to \{0,1\}$ such that any circuit computing f has size $\Omega(2^n/n)$.

Solution. Let $\mathcal{F} = \left\{ f : \{0,1\}^n \to \{0,1\} \right\}$ be the set of functions from $\{0,1\}^n$ to $\{0,1\}$. Note that $|\mathcal{F}| = 2^{2^n}$. On the other hand, for a size $s \in \mathbb{N}$, let $\mathcal{C}_s$ be the set of circuits of size s which have n input gates. Since a circuit is fully specified by selecting each gate and selecting the origins of the input wires for each gate, and since each gate takes (at most) 2 inputs, the total number of circuits of size s is at most $(5s^2)^s$ (the 5 is because there are 5 types of gates). Thus $|\mathcal{F}| = 2^{2^n}$ and $|\mathcal{C}_s| = 2^{\mathcal{O}(s \log s)}$. Clearly whenever s is such that $|\mathcal{C}_s| < |\mathcal{F}|$, then there must be some $f \in \mathcal{F}$ which cannot be computed by a circuit of size s. For some constant c, setting $s = c2^n/n$ gives $s \log s = c2^n/n \cdot \big(n - \log(n) + \log(c)\big) < c2^n$, and so

$$|\mathcal{C}_s| = 2^{\mathcal{O}(s \log s)} < 2^{2^n} = |\mathcal{F}|.$$

Thus, there exists some $f \in \mathcal{F}$ which requires a circuit of size $\Omega(2^n/n)$.