1 Reducing the randomness of repeated runs

Assume we have a randomized algorithm $A$ using an $r$ bit random string $R$ that approximates a function $f$ with error $\frac{1}{100}$, namely, $\Pr_{R}[A(x, R) \neq f(x)] \leq \frac{1}{100}$. To improve the error bound, we run $A$ repeatedly and take the majority output. However, if we were to do this naively, then running $A$ $k$ times would take $rk$ random bits; here we show how to accomplish this using only $r + O(k)$ random bits.

The construction involves the following: we consider a graph $G$ on $2^r$ nodes of a special form; we wish to simulate drawing $k$ random vertices from the graph using less than $rk$ bits of randomness; we accomplish this via a random walk on $G$.

Specifically, let $G$ have the following properties:

- $G$ is $d$-regular for some fixed $d$, i.e., every vertex of $G$ has degree $d$.
- Let $P$ be the transition matrix for a random walk of $G$; we require $P$ be symmetric, and that $\lambda_2$, the second largest eigenvalue of $P$, have magnitude at most $\frac{1}{10}$.

We define the following algorithm:

1. Pick a random start node $R \in \{0, 1\}^r$.

2. Repeat the following $7k$ times:
   
   (a) Let $R$ be a random neighbor of the old $R$
   
   (b) Run $A(x)$ with $R$ as the random bits

3. Output the majority answer

We note that each of the $7k$ iterations requires choosing one of the $d$ neighbors of the old $R$, and thus requires $\log d$ bits of randomness. Thus the above algorithm requires $r + 7k \log d = r + O(k)$ random bits, since we take $d$ to be a constant. We note that this is significantly less than the $O(rk)$ bits of randomness required by the naive algorithm.

We prove the following:

**Theorem 1** Under the above assumptions, the above algorithm will output $f(x)$ with error at most $2^{-k}$.

To help our analysis of the above theorem, we define the set $B$, the “bad guys” as follows for some fixed $x$:

$$B = \{ R | A_R(x) \text{ is incorrect} \},$$

namely the set of $R$ for which $A_R(x) \neq f(x)$. We note that by definition of $A$, $|B| < 2^r/100$.

We next define two diagonal matrices $N, M$ of size $2^r \times 2^r$. Let $N$ have a 1 on the diagonal entry $(R, R)$ for each string $R \in B$, and zeros everywhere else. Let $M$ have a 1 on the diagonal entry $(R, R)$ for each string $R \notin B$, i.e., $M = I - N$.

For a vector $v$ let $|v|$ denote the $L_1$ norm of $v$, namely $\sum_i |v_i|$, and let $||v||$ denote the $L_2$ norm of $v$, namely $\sqrt{\sum_i v_i^2}$.

Consider the following constructions. Let $p$ be a probability distribution on the strings of length $r$. Then

$$|pN| = \Pr_{R \sim p} [R \text{ is bad}].$$
Similarly we have

\[ |pPN| = Pr_{R \leftarrow p} [\text{start at } p, \text{ take one step according to } P, \text{ and end up in a bad } R], \]

and

\[ |pPNPN| = Pr_{R \leftarrow p} [\text{start at } p, \text{ take two steps according to } P, \text{ and end up in two bad nodes}]. \]

In general, given a "correctness path" \( S \), namely a sequence of "correct" or "incorrect" of length \( 7k \), if we let

\[ Q_i = \begin{cases} M & \text{if } S_i = \text{"correct"} \\ N & \text{if } S_i = \text{"incorrect"} \end{cases} \]

then the probability that our path through the graph will follow the "correctness path" \( S \) is

\[ Pr[S] = |p(PQ_1)(PQ_2) \cdot \ldots \cdot (PQ_{7k})|. \]

We state a lemma that will let us prove our theorem; we prove the lemma later.

**Lemma 2** For all distributions \( \Pi, \) and \( P, N, M \) as above

1. \( ||\Pi P M|| \leq ||\Pi|| \)
2. \( ||\Pi P N|| \leq \frac{5}{2}||\Pi|| \)

We now prove the main theorem.

**Proof of Theorem 1** Consider an execution of the above algorithm. If fewer than \( \frac{7k}{2} \) of the runs of \( A \) have randomness \( R \in B \) then a majority of the runs will output \( f(x) \) and the algorithm will output \( f(x) \) correctly. We bound the probability that this does not occur.

Consider a correctness path \( S \) which contains more than \( \frac{7k}{2} \) "incorrect"s. From above we have that

\[ Pr[S] = |p(PQ_1)(PQ_2) \cdot \ldots \cdot (PQ_{7k})|. \]

We have from the Cauchy-Schwarz inequality that for any vector \( v \) of length \( 2^r \)

\[ |v| = v \cdot (1, 1, ..., 1) \leq ||v|| \cdot ||(1, 1, ..., 1)|| = \sqrt{2^r}||v||. \]

Thus we have

\[ Pr[S] \leq \sqrt{2^r}||p(PQ_1)(PQ_2) \cdot \ldots \cdot (PQ_{7k})||. \]

At this point, we invoke Lemma 2 \( 7k \) times to successively remove the terms \( (PQ_i) \) from the above expression. We note that at least \( \frac{7k}{2} \) times we can invoke case two of the lemma. We thus have the bound

\[ Pr[S] \leq \sqrt{2^r}||p|| \left( \frac{1}{5} \right)^{7k/2}. \]

We note that the algorithm specified that the initial \( R \) be drawn randomly, and hence \( p \) is uniform. By explicit computation we may check that in this case

\[ ||p|| = \sqrt{\sum_{i=1}^{2^r} (2^{-r})^2} = \sqrt{2^{-r}}. \]

Thus \( Pr[S] \leq 5^{-7k/2}. \) We apply the union bound to bound the total probability of such a sequence \( S \) fooling the algorithm. The total number of sequences \( S \) is \( 2^{7k} \), and this thus bounds the number of
sequences with at least \( \frac{7k}{2} \) “incorrect” s in them. Thus the total probability of the algorithm giving the wrong answer is at most
\[
5^{-7k/2}2^{7k} = \left( \frac{4}{5} \right)^{7k/2} \leq 2^{-k},
\]
and we have the desired result. \( \blacksquare \)

We now prove the lemma.

**Proof of Lemma 2**

We note that the first part of the lemma, that \( ||P M P|| \leq ||P|| \) for any distribution \( P \) is trivially true since both \( P \) and \( M \) have all their eigenvalues at most 1. We write this out in greater detail.

For any vector \( x \),
\[
||x M|| = \sqrt{\sum_{i \in B} x_i^2} \leq \sqrt{\sum_i x_i^2} = ||x||.
\]
Thus \( ||P M P|| \leq ||P|| \).

Consider the eigenvalues \( \{\lambda_i\} \) and eigenvectors \( \{v_i\} \) of matrix \( P \). Since \( P \) is stochastic, it has an eigenvalue \( \lambda_1 = 1 \) with corresponding eigenvector of \( v_1 = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, ..., \frac{1}{\sqrt{2}}) \).

Recall that for a symmetric matrix, the eigenvectors form an orthonormal basis. Thus for any \( \Pi \) we can express it as
\[
\Pi = \sum \alpha_i v_i,
\]
for some \( \{\alpha_i\} \).

Thus we have
\[
||\Pi P|| = ||\sum \alpha_i v_i P|| = ||\sum \alpha_i \lambda_i v_i||,
\]
where the last equality is by from the definition of eigenvectors. Since the eigenvectors are orthonormal, we express this norm as
\[
||\sum \alpha_i \lambda_i v_i|| = \sqrt{\sum \alpha_i^2 \lambda_i^2} \leq \sqrt{\sum \alpha_i^2},
\]
where this last inequality is because \( \lambda_i \leq 1 \) by assumption. Recall that we defined \( \{\alpha_i\} \) by \( \Pi = \sum \alpha_i v_i \), so since \( \{v_i\} \) is an orthonormal basis we have
\[
\sqrt{\sum \alpha_i^2} = ||\Pi||,
\]
from which we conclude that \( ||P M P|| \leq ||\Pi|| \), as desired.

We now turn to the second part of the lemma, that \( ||P P N|| \leq \frac{4}{5}||\Pi|| \). Similar to the above analysis, we have
\[
||P P N|| = ||\sum \alpha_i v_i P N|| = ||\sum \alpha_i \lambda_i v_i N|| \leq ||\alpha_1 \lambda_1 v_1 N|| + ||\sum_{i=2}^{2^r} \alpha_i \lambda_i v_i N||,
\]
where the last inequality is by the triangle inequality. We bound each of these terms separately.

Consider the first term, \( ||\alpha_1 \lambda_1 v_1 N|| \). Since \( ||\{\alpha_i\}|| = ||\Pi|| \) we have \( \alpha_1 \leq ||\Pi|| \). From above we have that \( \lambda_1 = 1 \) and \( v_1 = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, ..., \frac{1}{\sqrt{2}}) \). Since \( N \) has a one on its diagonal for each \( R \in B \) we have
\[
||\alpha_1 \lambda_1 v_1 N|| \leq ||\Pi|| \cdot ||v_1 N|| = ||\Pi|| \cdot \sqrt{\sum_{i \in B} 2^{-r} \leq ||\Pi|| \sqrt{\frac{1}{100}} = \frac{||\Pi||}{10}.}
\]

We now bound the second term: \( ||\sum_{i=2}^{2^r} \alpha_i \lambda_i v_i N|| \). Since \( N \) is a diagonal matrix, each of whose entries is at most 1, we have \( ||\sum_{i=2}^{2^r} \alpha_i \lambda_i v_i N|| \leq ||\sum_{i=2}^{2^r} \alpha_i \lambda_i v_i|| \). Since the vectors \( \{v_i\} \) form an orthonormal
basis, we have \( ||\sum_{i=2}^{2r} \alpha_i \lambda_i v_i|| = \sqrt{\sum_{i=2}^{2r} \alpha_i^2 \lambda_i^2} \). Since by hypothesis all the eigenvalues of \( P \) except the first have magnitude at most \( \frac{1}{10} \), we have

\[
\sqrt{\sum_{i=2}^{2r} \alpha_i^2 \lambda_i^2} \leq \frac{1}{10} \sqrt{\sum_{i=2}^{2r} \alpha_i^2} \leq \frac{||\Pi||}{10},
\]

our desired bound.

Summing these two bounds, we conclude \( ||\Pi PN|| \leq \frac{||\Pi||}{5} \), as desired.

2 Derandomizing

We have just seen a technique for reducing the randomness needed for an algorithm. We ask now: what techniques might completely eliminate randomness from an algorithm?

The most basic such technique is the enumeration technique, which is just:

1. Given an algorithm \( A \) that uses \( r \) uniformly chosen random bits and succeeds with probability more than \( \frac{1}{2} \),
2. Run \( A \) \( 2^r \) times for every possible \( r \)-bit string \( R \).
3. Output the majority answer

Clearly the resulting algorithm uses no randomness, and outputs the correct answer. However, its running time is \( 2^r \) times longer than \( A \), which might be prohibitive.

We sketch an alternative that is applicable when \( A \) uses its random bits in a very particular way.

Recall:

**Definition 3 (Independent)** \( R_1, R_2, \ldots, R_n \in T \) are independent if for all \( b_1, b_2, \ldots, b_n \in T^n \),

\[
\Pr[R_1R_2\ldots R_n = b_1b_2\ldots b_n] = |T|^{-n}.
\]

Values chosen uniformly at random are independent.

Often, this is more than we need, and pairwise independence is sufficient.

**Definition 4 (Pairwise independent)** \( R_1, R_2, \ldots, R_n \in T \) are pairwise independent if for all \( i \neq j \in [1, n] \), for all \( b_i, b_j \in T^2 \),

\[
\Pr[R_iR_j = b_i b_j] = |T|^{-2}.
\]

Intuitively this means that any pair of bits of \( R_i, R_j, i \neq j \) will appear uniformly random. This notion may be extended to larger subsets of variables.

**Definition 5 (k-wise independent)** \( R_1, R_2, \ldots, R_n \in T \) are k-wise independent if for all \( i_1 < i_2 < \ldots < i_k \in [1, n] \) and \( b_{i_1}, b_{i_2}, \ldots, b_{i_k} \in T^n \),

\[
\Pr[R_{i_1}R_{i_2}\ldots R_{i_k} = b_{i_1}b_{i_2}\ldots b_{i_k}] = |T|^{-k}.
\]

As an example of a pairwise random distribution of 3-bit strings, consider the uniform distribution over the strings \( \{000, 011, 101, 110\} \), and note that for any pair of bits, all four possibilities appear exactly once. Also note that the 3rd bit is the exclusive-or of the first two.

Suppose we have an algorithm \( A \) that uses \( r \) random bits, but only requires pairwise independence, instead of full independence. Further, suppose we have a generator \( G \) that, when given \( m \ll r \) fully random bits outputs \( r \) pairwise random bits. Then we have the following procedure:

For each of the \( m \)-bit strings \( M \), run the generator on \( M \) and let \( R = G(M) \). Then run \( A \) with \( R \) as the random bits. Output the majority answer that these runs of \( A \) return.
We note that since $A$ only requires pairwise independent bits, the above procedure – a trivial modification of the enumeration technique above – will output the correct answer, and require time only $2^m$ more than the time taken by $A$ and $G$.

This approach relies on two facts which we will see in later lectures:

- A variety of algorithms do not in fact require “complete” independence, and only require pairwise independence, or other weaker notions of independence.

- There exist very efficient generators for producing pairwise, 3-wise, etc. independent strings from much shorter (polylogarithmic) fully independent strings.