1 Review of Important Probability Bounds

Let $X_1, ..., X_n$ be 0/1 random variables with common probability p. Let $S = \sum_{i=1}^{n}$. We have the following two bounds on deviations away from S.

Chernoff Bounds: For n independent variables, we have for $0 < \delta < 1$:

$$\Pr_{X_1,\ldots,X_n}[|S_n - pn| > \delta n] \leq 2e^{-\frac{\delta^2 n}{2}}$$

Note that pn is the same as the expected value of S. Essentially, this bound says that probability the sum of n independent variables deviates from the expected value by a factor of δn decreases exponentially as n increases. (We do not even use it in the proof)

Chebyshev Bound: For n pair-wise independent variables, we have for $0 < \delta < 1$:

$$\Pr_{X_1,\ldots,X_n}[|S_n - pn| > \delta n] \leq \frac{1}{4\delta^2 n}$$

In other words, the probability that S deviates from the expected value decreases linearly as n increases.

2 Finishing Proof for Goldreich-Levin

Recall that the goal is proving that $< x, r >$ for a random r, x is a hard-core predicate given $(r, f(x))$ for one way function f. This means that no poly-time adversary can successfully guess $b(x, r)$ given $(f(x), r)$ (where b takes and returns the dot product mod 2. Formally:

$$\forall A, \forall c \; Pr_{x,r \leftarrow \$}[A(f(x), r) = b(x, r)] \leq \frac{1}{2} + \frac{1}{n^c}$$

The proof aims to demonstrate that if $b(x, r)$ was not hard-core predicate, then there would exist an adversary that could invert f with non-negligible probability. Consider an adversary A_b that could predict $b(x, r)$ given $(f(x), r)$ with a non-negligible advantage. This adversary is used as an oracle that will be used to eventually invert f.

2.1 Generating Samples

2.1.1 Proof of Correctness

Firstly, we prove the following claim:

$$< x, s_1 \oplus s_2 > = < x, s_1 > \oplus < x, s_2 >$$

In otherwords, if we have valid pairs (r_i, b_i) and (r_j, b_j), we can produce a new pair $(r_i \oplus r_j, b_i \oplus b_j)$ that is also correct. For notation purposes, we assume $|x| = |r| = n$. To prove the previous claim, we need to show that:

$$(\sum_{i=1}^{n} r_1^i x^i) \ mod\ 2 \oplus (\sum_{i=1}^{n} r_2^i x^i) \ mod\ 2 = (\sum_{i=0}^{n} (r_1^i \oplus r_2^i) x^i) \ mod\ 2$$

Recall that the $\oplus$ operator is simply addition modulo 2 over each bit, and that the modulo operator is distributive. Therefore, we can rewrite the leftside of the equation as:

$$((r_1^1 x^1 + ... + r_1^n x^n) \ mod\ 2 + (r_2^1 x^1 + ... + r_1^n x^n) \ mod\ 2) \ mod\ 2$$
$$(\sum_{i=1}^{n} (r_1^i x^i + r_2^i x^i)) \ mod\ 2$$
$$(\sum_{i=1}^{n} (r_1^i + r_2^i) x_i) \ mod\ 2$$

Using once again the fact that *mod* is distributive over multiplication and addition we have:

$$(\sum_{i=1}^{n} (r_1^i + r_2^i) \ mod\ 2 \cdot x^i \ mod\ 2) \ mod\ 2$$

Finally, since x^i is 0 or 1, $x^i \ mod\ 2 = x^i$ and we have:

$$(\sum_{i=1}^{n} (r_1^i \oplus r_2^i) x^i) \ mod\ 2$$

Completing the proof.

2.1.2 Constructing R

To successfuly invert f, we need to generate a sufficient number of samples of strings r and bits b such that:

$$r_i \oplus x = b_i$$

Trivially, we could produce m random strings r and guess whether each $r_i \oplus x = 0/1$. However, the probability of successfully guessing all m b_i values correctly is $1/2^n$, which is negligibly probable. However, if we can successfully guess b for $log(m)$ r's, we can construct $2^{log(m)} = m$ r's that are pairwise independent by xoring all subsets of the $log(m)$ r values we generated by using the fact we proved above. The probability of guessing $log(m)$ bits is $1/m$ which is non-negligible and therefore can be used in our algorithm.

2.2 Generating x

The plan to invert $f(x)$ is as follows:

1. For each index $j = 1, ..., n$, for every r_i, b_i pair, take $A_b(r_i^j, f(x))$.
2. Take the output $\hat{b_i}$ and find $\hat{b_i} \oplus b_i$
3. The ith bit of x is assumed to be the majority value of the previous step accross all $i = 1, ..., m$.

To demonstrate that this will output x with a non-negligible probability, we use Chebyshev bound with $S = \sum_{i=1}^{n} f(i)$, where $f(i) = 1$ if the ith guessed bit is wrong else 0, to get:

$$\begin{aligned} Pr[S > \frac{1}{2}] &= Pr[|S - mp| > \frac{m}{2} - mp] \\ &= Pr[|S - mp| > (\frac{1}{2} - p)m] \\ &\leq \frac{1}{4[(\frac{1}{2} - p)^2]m} \end{aligned}$$

Since p is the probability that A guesses incorrectly, $\frac{1}{2} - (\frac{1}{2} - \epsilon(n)) = \epsilon(n)$ (Note: since we assume x is good the advantage is technically $\epsilon(n)/2$, however, we just use $\epsilon(n)$ for convenience). Therefore we can rewrite the probability as:

$$\leq \frac{1}{4[\epsilon(n)^2]m}$$

Therefore, probability that x_i is wrong is less than the above inequality for all i. By the union bound, we have:

$$\bigcup_{i=1...n} Pr[x_i \text{ is wrong}] \leq \frac{n}{4[\epsilon(n)^2]m}$$

Which means that our algorithm works with probability:

$$\geq 1 - \frac{n}{4[\epsilon(n)^2]m}$$

To get a non-negligible probability, we set $m = \frac{n}{2\epsilon(n)^2}$ to guarantee this inversion works with probability $\geq 1/2$.

2.2.1 Notes

Generating m Bits for Given m Recall that to generate m (r_i, b_i) pairs, we need to guess $log(m)$ bits. Doing this for $m = \frac{n}{2\epsilon(n)^2}$ suceeds with probability:

$$\frac{2\epsilon(n)^2}{n}$$

Since epsilon is non-negible, and multiplying a non-negligible function by another non-negligible function $(1/n)$ results in a non-neglible function, the probability of guessing $log(m)$ bits is non-negible.

Total Probability To demonstrate the entire algorithm happens with non-negligible probability, we rewrite it as the product of event probabilities. For all probabilities, we use the lower bound found in each step.

$$Pr[x \text{ is good}] \cdot Pr[\text{Correctly generate } m \; r\text{'s}] \cdot Pr[\text{All } x\text{'s are correct}]$$
$$= \frac{\epsilon(n)}{2} \cdot \frac{2\epsilon(n)^2}{n} \cdot \frac{1}{2}$$

Since the product of negible functions are negligible, our algorithm suceeds with non-negligible probability given A_b has an advantage.

3 Pseudo Random Generator (PRG)

Pseudo random generators are functions that take in a short, truly random seed s and generates an output of poly-many bits $R(s)$ such that the output looks random. We say that a PRG R is secure if it is indistinguishable from a truly random string if its output on random seed s cannot be differentiated by any polynomial time adversary. Formally:

$$\forall A \in PPT, \forall c, |Pr[\underset{s \leftarrow \$}{R(s)} = 1] - Pr[\underset{r \leftarrow \$}{A(r)} = 1]| < \frac{1}{n^c}$$

This is simulated in a game of sorts where A is passed a string that was either generated completely randomly, or generated from random seed s and PRG R. We say that the probability that A can guess which of these two events happened is less that $1/2 + negl$.

3.1 Example PRG

We have the following PRG that can take an input x and produces outputs of length $|x|+1$ using one way function f. It uses a fixed random r and outputs

$f(x)| < x, r >$. The reason this is a PRG is because if an adversary A had an advantage over R, it would also have an advantage over the hardcore-predicate function, which we proved is not possible.

3.2 Computational indistinguishability

Two probability distributions are **computationally indistinguishable** if no poly-time adversary can distinguish stream samples from either. More formally, two probability distributions X, Y are **computationally indistinguishable** if:

$$\forall A \in PPT, \forall c, |\underset{x \leftarrow X_n}{Pr}[A(x) = 1] - \underset{y \leftarrow Y_n}{Pr}[A(y) = 1]| < \frac{1}{n^c}$$

3.2.1 Showing Single Sample Advantage

Claim. Computational indistinguishability over polynomially many samples $\iff$ computational indistinguishability over one sample.

We show that if there exists an A that has a non-negligble advantage $\epsilon(n)$ given m samples, then there exists an A' that has a non-neglible advantage $\epsilon(n)/m$ on a single sample.

Consider A that either gets $x_1, ..., x_n$ from X or $y_1, ..., y_n$ from Y and can distinguish between the two with $\epsilon(n)$ advantage. Firstly, observe that for this to be the case, then with $Pr[A(x_1, ...x_n) = 1] = p$ and $Pr[A(y_1, ..., y_n) = 1] = q$, we have $p - q = \epsilon(n)$. Now, we define n adversaries A_i that independently samples from Y and X and returns:

$$A(y_1, ..., y_{i-1}, x_i, ..., x_n)$$

Consider each A_i which outputs 1 with probability p_i. We have the following summation:

$$\sum_{i=1}^{n-1}(Pr[A_i = 1] - Pr[A_{i+1} = 1]) = p - q$$

The claim is that there exists at least one (A_i, A_{i+1}) pair where $p_i - p_{i+1}$ is at least $\epsilon(n)/m$. The proof is by contradiction, assume that all $p_i - p_{i+1}$ were less that $\epsilon(n)/m$. The the above summation would be strictly less that $\epsilon = p - q$ showing the assumption is false. This means that for that A_i, A_{i+1} pair, we can construct A' as follows.

1. Randomly sample i y's from Y and $n - i - 1$ x's from X.

2. Return $A(y_1, .., y_i, \mu, x_{i+1}, ..., x_n)$

If μ was sampled from X, then it simulates A_i, if sampled from Y then it simulates A_{i+1}. This gives us:

$$|\underset{x \leftarrow X}{Pr}[A(x) = 1] - \underset{y \leftarrow Y}{Pr}[A(y) = 1]| \geq \frac{\epsilon(n)}{m}$$

Demonstrating the claim.