Lectures on 8 and 13 July 1998

April 26, 1998

Chernoff Bound.

Let $y \in \{0,1\}^m$. Pick m times independently with uniform probability, $x_i \in \{0,1\}$ for $i \in \{1 \ldots m\}$.

Define the random variable $z_i = \begin{cases} 1 & \text{if } x_i = y_i \\ -1 & \text{if } x_i \neq y_i \end{cases}$

Let $\mathcal{Z}_m$ be the random variable $\mathcal{Z}_m = \sum_{i=1}^m z_i$.

$\mathcal{Z}_m$ is a sum of m independent random variables, then $E(\mathcal{Z}_m) = E(\sum_{i=1}^m z_i) = \sum_{i=1}^m E(z_i) = \sum_{i=1}^m 0 = 0$.

Theorem 1 *Prob*$(\mid \mathcal{Z}_m \mid \geq b) < \frac{2}{e^{\frac{b^2}{2^m}}})$ *for any* $b > 0$. □

Theorem 2 $BPP \subseteq NP^{NP}$

Proof.

We use $\mathrm{NP}^{\mathrm{NP}} = \Sigma_2^p$ and the following characterization of the Polynomial Time Hierarchy by Alternating Turning Machines: $\Sigma_k^p = \mathrm{A}\Sigma_k^p$ ($\Pi_k^p = \mathrm{A}\Pi_k^p$).

The crux of the proof is an $\mathrm{A}\Sigma_2^p$ algorithm which simulates a given probabilistic polynomial time algorithm. The crux for the correctness of the $\mathrm{A}\Sigma_2^p$ algorithm is the existence of a k-pseudorandom generator g. And the existence of a $2^{\epsilon m}$-hard function (cf. homework) f (g is defined using f) is the crux for g being a k-pseudorandom generator. (i.e., if g is not a k-pseudorandom generator f will not be $2^{\epsilon m}$-hard)

Roughly speaking, picking up an element at random is choosing without any help from the structure of the set of elements. And a hard function is one "without structure", the best we can do, almost, is answer at random. It is intuitive then to try to build a random sequence from a hard function. Consider a behaviourist definition of structure and randomness if we are in a resource bounded realm.

A pseudorandom generator is, intuitively, a way to expand a random seed into a sequence "random enough" for a bounded time algorithm. To our purpose, we do an ad hoc definition of a pseudorandom generator.

Definition 1 (r-Pseudorandom Generator) *Let D_k be a probability distribution on Σ^k and let R_k the uniform probability distribution on Σ^k, i.e., $R_k(x) = 2^{-k}$ for all $x \in \Sigma^k$.*

Let G be a function $G : \{0,1\}^{c \log n} \to \{0,1\}^{n^k}$ for any integer $c, k, n \geq 1$.

Let D^G the probability distribution on Σ^{n^k} induced by $G(x)$ when x is chosen at random from $R_{c \log n}$.

G is a r-pseudorandom generator iff for any family of circuits $\{C_n\}$ of size bounded by n^{2r} and any $X \in \Sigma^n$,

$$\mid Prob_Y(C(X,Y) = 1) - Prob_Z(C(X, G(Z)) = 1) \mid \leq 1/2 - 1/n^r$$

for all sufficiently large n. Y and Z are chosen from R_{n^k} and $R_{c \log n}$ respectively. □

Let $A \in$ BPP and let M be a probabilistic polynomial time algorithm that decides A in BPP. Without loss of generality let n^k be the running time of M and $1/2^n$ the error bound of M where n is the input size. Let M' be the "deterministic version" of M where the coin flips results are given as a second input of length n^k.

We give an $\mathbf{A}\Sigma_2^p$ algorithm $M_{\mathbf{A}\Sigma_2^p}$ for A:

input (x);
for some integer $a > 0$, integer $b > a$, real $\epsilon > 0$;
comment: *a will be fixed at the end of the proof. b depends on a. ϵ is such that there is a $2^{\epsilon m}$-hard function.*
define constant $n = \mid x \mid$, $m = a \log n$;
define variable
$f : \{0,1\}^{a \log n} \to \{0,1\}$;
$g : \{0,1\}^{b \log n} \to \{0,1\}^{n^k}$;
define algorithm M' defined in the statement above;

begin
branch-$\exists$ and then guess(f);
branch-$\forall$ and then check that f is $2^{\epsilon m}$-hard;

comment: *Note that each branch will be associated with a possible circuit and will compute that circuit for all values of length m to decide if the circuit approximates the function.*
comment: *Note that from this point on the algorithm is in P.*
Use f to build a k-pseudorandom generator g;
Run $M'(x, g(y))$ for all $y \in \{0,1\}^{b\log n}$;
accept iff at least half of the computations $M'(x, g(y))$ accept.
comment: *Note that the probability of M being right can differ from 1 by no more than on exponentially small amount and a k-pseudorandom generator varies this probability no more than $1/2 - 1/n^k$, thus, for large enough n the rate of acceptance of M' can not go across 1/2. i.e., if Prob(M accepts) is almost 1 (0) then Prob(M' accepts) will be over (under) 1/2.*
end.

Let C' be the circuit that simulate the Turing Machine M'. Remember that DTIME(n^k) $\subseteq$ SIZE(n^{2k}).

Proof of correctness of the algorithm.

Since g is a k-pseudorandom generator, by definition, for all $X \in \Sigma^n$ the following holds

$$| \text{Prob}_Y(C'(X,Y) = 1) - \text{Prob}_Z(C'(X, g(Z)) = 1) | \quad \leq \quad 1/2 - 1/n^k$$

for all sufficiently large n. Y and Z are chosen from R_{n^k} and $R_{b\log n}$ respectively.

And since M decides A in BPP,

$$\begin{aligned} X \in A \quad &\Rightarrow \quad \text{Prob}_Y(C'(X,Y) = 1) \geq 1 - 1/2^n \\ X \notin A \quad &\Rightarrow \quad \text{Prob}_Y(C'(X,Y) = 1) \leq 1/2^n \end{aligned}$$

Then, (assume that if $X \in A$ [$X \notin A$] then $\text{Prob}_Y(C'(X,Y) = 1) > \text{Prob}_Z(C'(X, g(Z)) = 1)$ [$\text{Prob}_Y(C'(X,Y) = 1) < \text{Prob}_Z(C'(X, g(Z)]$. Note that is only necessary to prove this case)

$X \in A \quad \Rightarrow \quad \text{Prob}_Z(C'(X, g(Z)) = 1) \geq 1 - 1/2^n - 1/2 + 1/n^k = 1/2 - 1/2^n + 1/n^k$
$> 1/2$ for sufficiently large n

$X \notin A \quad \Rightarrow \quad \text{Prob}_Z(C'(X, g(Z)) = 1) \leq 1/2^n + 1/2 - 1/n^k = 1/2 + 1/2^n - 1/n^k$
$< 1/2$ for sufficiently large n $\square$

Proof that g is a k-pseudorandom generator.

Fact 1 *Let n, k, a and b denote the same values as in the algorithm $M_{A\Sigma_2^p}$.*

There exists a polynomial time algorithm with input 1^n that outputs a matrix N of n^K rows and $b \log n$ columns. Furthermore, each row has $a \log n$ 1's and for any two rows i and j, if we let $C_i = \{$columns with value 1 in row number $i\}$, then $\mid C_i \cap C_j \mid \leq \log n$. $\square$

We will prove it in the next lecture.

Algorithm to compute g:

input (y);
define n, k, a, b, matrix A defined in Fact 1;
{it holds: $\mid y \mid = b \log n$**}**
define set $C_{1 \ldots n^k}$ defined in Fact 1;
define function f defined in algorithm $M_{\mathrm{A}\Sigma_2^p}$;

begin
for $i = 1$ **to** n^k;
    {goal: compute the ith bit of $g(y)$**}**
    Get C_i;
    Let projection(y, C_i) be the subsequence in y obtained picking
        the bits of y corresponding to elements in C_i;
    {it holds: $|\text{projection}(y, C_i)| = a \log n$**}**
    add to output $f(\text{projection}(y, C_i))$;
end.

We know that f is $2^{\epsilon m}$-hard.

Claim 1 *If f is $2^{\epsilon m}$-hard then g is a k-pseudorandom generator.*

Notice that m, k and g are the values defined in the algorithm $M_{\mathrm{A}\Sigma_2^p}$.

We prove it by contradiction. Suppose that g is not a k-pseudorandom generator. Then it holds that exists a family of circuits $\{D_n\}$, where SIZE(D_n) $\leq n^{2k}$, and a $X \in \Sigma^n$,

$$\mid \text{Prob}_Y(D(X, Y) = 1) - \text{Prob}_Z(D(X, g(Z)) = 1) \mid \quad > \quad 1/2 - 1/n^k \quad (1)$$

for infinitely many n. Y and Z are chosen from R_{n^k} and $R_{b \log n}$ respectively.

We introduce some notation. If s is a string of length k, let s_i denote the ith bit of s and let $s_{[i...j]}$ denote the bits from i through j of s, where $1 \leq i \leq j \leq k$.

Let p_i denote $\text{Prob}_{V,W}(D(X, g(V)_{[1...i]}, W) = 1)$ where W and V are chosen from R_{n^k-i} and $R_{b \log n}$ respectively.

Note that $p_0 = \text{Prob}_Y(D(X, Y) = 1)$ and $p_{n^k} = \text{Prob}_Z(D(X, g(Z)) = 1)$.
We rewrite the equation (1) as $\mid p_0 - p_{n^k} \mid > 1/2 - 1/n^k$.
As we "go" from p_0 to p_{n^k} in n^k steps there exist one step i such that

$$\mid p_i - p_{i+1} \mid > (1/2 - 1/n^k)/n^k > 1/n^{d'}$$

for some d'.

$\mid p_i - p_{i+1} \mid$ is the average over all possible choices of Z and Y. We can fix some election of any set of bits and then $\mid p_i - p_{i+1} \mid$ is the average over all possible ways of complete Z and Y. Therefore exists some string $Y' \in \Sigma^{n^k-i-1}$ and some string $Z' \in \Sigma^{b \log n - a \log n}$ such that $\mid p_i - p_{i+1} \mid > 1/2 - 1/n^{d'}$ where $\mid p_i - p_{i+1} \mid$ is the average over the probabilities obtained by choosing at random the $a \log n$ bits, let's name that Z'', in C_{i+1} (remind that we defined C_i from the matrix A) that complete Z and one bit, Y'', to complete Y.

Let D' be the circuit so that

$$D'(Z'', Y'') = D(X, g(Z)_1, g(Z)_2 \ldots g(Z)_i, Y'', Y'_1, Y'_2 \ldots Y'_{n^k-i-1})$$

Note that Z is a mixture of Z' and Z'' and Y the concatenation of Y'' and Y'. Note also that X, Z' and Y' are fixed.

Later we also fix Y'' but for now let it random.

So, first our circuit has to compute the first i components of $g(Z)$. It means i circuits for the f function where the input size is at most $\log n$ (remember that $\mid C_i \cap C_j \mid \leq \log n$). Calculating its CNF entails a size bounded by $i2n \log n$. Then, we already have all the inputs available to compute D'. We know that the size of D is bounded by n^{2k}.

Therefore the circuit D' has size bounded by $i2n \log n + n^{2k}$ for inputs of size $a \log n$.

Take in account that we consider Y'' taking values at random.

Let D'' the circuit which compute the following function:

$$D''(Z'', Y'') = \begin{cases} Y'' & \text{if } D'(Z'', Y'') = 1 \\ \text{not } Y'' & \text{if } D'(Z'', Y'') = 0 \end{cases}$$

D'' has size bounded by size$(D') + t$, for some little constant t. That equals $i2n\log n + n^{2k} + t$.

Let's see how well D'' computes f. Since Y'' is chosen at random, it is equally likely to be $f(Z'')$ as not to be. The probability that D'' computes f is one-half the probability that a correct Y'' is maintained plus one-half the probability that a wrong Y'' is changed.

Let

$$q = \text{Prob}_{Z'',Y''}(D(X, g(Z)_1, g(Z)_2 \ldots g(Z)_i, \overline{g(Z)_{i+1}}, Y'_1, Y'_2 \ldots Y'_{n^k-i-1}) = 1)$$

Notice that $p_i = (p_{i+1} + q)/2$.

Let's say that D'' is correct for an input whenever D'' computes f for that input. Then

$$\begin{aligned}
\text{Prob}(D'' \text{ is correct}) &= \frac{1}{2}p_{i+1} + \frac{1}{2}(1-q) \\
&= \frac{1}{2}(p_{i+1} + (1 - (2p_i - p_{i+1})) \\
&= \frac{1}{2} + p_{i+1} - p_i \\
&> \frac{1}{2} + \frac{1}{n^{d'}}
\end{aligned}$$

Again for Y'' fixed we have a new probability and at least one of these two probabilities has to be as high as the average. Therefore we fix Y'' to this value.

Let D''' be the circuit D'' where the input Y'' is fixed to this value.

So far, under the assumption that g is not a pseudorandom generator we have a circuit D''' with $a\log n$ inputs (we will fix a later and n is one of the infinitely many n that make the equation (1) hold) such that

$$\begin{aligned}
\text{SIZE}(D''') &\leq 2in\log n + n^{2k} + t \\
\text{Prob}(D'''(x) = f(x)) &> \tfrac{1}{2} + \tfrac{1}{n^{d'}}
\end{aligned}$$

We show now that f is not $2^{\epsilon m}$-hard to obtain the contradiction required. We see that exists an input length, l, and a circuit T for that input length such that

$$\begin{aligned}
\text{SIZE}(T) &\leq 2^{\epsilon l} \\
\text{Prob}(T(x) = f(x)) &> \tfrac{1}{2} + \tfrac{1}{2*2^{\epsilon l}}
\end{aligned}$$

We claim that $l = a \log n$ and $T = D'''$ prove that f is not $2^{\epsilon m}$-hard:

$$\begin{aligned}
\text{SIZE}(D''') &\leq 2in \log n + n^{2k} + t \\
&\leq 2^{\epsilon m} = 2^{\epsilon a \log n} = n^{\epsilon a} \text{ for all sufficiently large } a \\
\text{Prob}(D'''(x) = f(x)) &> \frac{1}{2} + \frac{1}{n^{d'}} \\
&\geq \frac{1}{2} + \frac{1}{2 * 2^{\epsilon a \log n}} = \frac{1}{2} + \frac{1}{2n^{\epsilon a}} \text{ for all sufficiently large } a
\end{aligned}$$

Fix then a as required and get the contradiction required. □