Relativized Classes and Probabilistic Classes

We will now consider some classes related to NP, relativized classes, Probabilistic classes, and randomized algorithms for some interesting problems.

Let $\overline{L}$ denote the complement of L.

$\mathbf{coNP} = \{L : \overline{L} \in \mathbf{NP}\}$.

In general for any of the class $\mathbf{C}$, we define $\mathbf{coC} = \{L : \overline{L} \in \mathbf{C}\}$.

Consider the problem of primality. (That is: given a number n, in binary, is n a prime?).

The complement problem of primality, is composite.

Easy: composite problem is in **NP** (just guess and verify a factor q of n, such that $1 < q < n$).

Thus, primality is in **coNP**.

It has recently been shown that primality is also in **P**.

Relativized classes

Suppose $\mathbf{C}$ is a class of languages. Then
$\mathbf{P}^{\mathbf{C}} = \{L : (\exists$ polynomial time oracle Turing Machine $M)(\exists L' \in \mathbf{C})[M^{L'}$ accepts $L]\}$.
$\mathbf{NP}^{\mathbf{C}} = \{L : (\exists$ polynomial time nondeterministic oracle Turing Machine $M)(\exists L' \in \mathbf{C})[M^{L'}$ accepts $L]\}$.

Polynomial Hierarchy

Let $\Sigma_0^p = \mathbf{P}$.
$\Sigma_1^p = \mathbf{NP}$.
$\Pi_1^p = \mathbf{coNP}$.
$\Sigma_{i+1}^p = \mathbf{NP}^{\Sigma_i^p}$.
$\Pi_{i+1}^p = \{L : \overline{L} \in \Sigma_{i+1}^p\}$.
It can be shown that
$L \in \Sigma_i^p$ iff there exists a polynomial time decidable predicate $R(x, y_1, y_2, \ldots, y_i)$ such that
$x \in L \Leftrightarrow (\exists y_1) \ldots (Qy_i)[R(x, y_1, y_2, \ldots, y_i)]$, where the quantifiers are polynomially bounded.
Similarly, $L \in \Pi_i^p$ iff there exists a polynomial time decidable predicate $R(x, y_1, y_2, \ldots, y_i)$ such that
$x \in L \Leftrightarrow (\forall y_1) \ldots (Qy_i)[R(x, y_1, y_2, \ldots, y_i)]$, where the quantifiers are polynomially bounded.

$\mathbf{P}^{\mathbf{NP} \cap \mathbf{coNP}} = \mathbf{NP} \cap \mathbf{coNP}$

Since $\mathbf{P}^{\mathcal{C}}$ is closed under complementation, and $\mathbf{NP} \cap \mathbf{coNP} \subseteq \mathbf{P}^{\mathbf{NP} \cap \mathbf{coNP}}$, it suffices to show that $\mathbf{P}^{\mathbf{NP} \cap \mathbf{coNP}} \subseteq \mathbf{NP}$.
Consider any polytime bounded oracle machine M, with oracle for $A \in \mathbf{NP} \cap \mathbf{coNP}$.
Let N_1 witness that $A \in \mathbf{NP}$ and N_2 witness that $\overline{A} \in \mathbf{NP}$.

$N_3(x)$:

- Simulate $M(x)$.
- Whenever, M asks a question w, N_3 guesses the answer. Based on whether it guesses the answer to be Yes, or No, it tries to verify the answer by simulating N_1 or N_2 respectively.
- If it cannot verify, then it halts rejecting the input.
- If it can verify the answer, it proceeds with the simulation of M.
- At the end of simulation of M, N_3 accepts, iff M does.

Clearly, N_3 is polytime bounded.
Also, N_3 doesn't accept any x not accepted by M.
For x accepted by M, there is a sequence of right guesses for the answers and right verification of the answers which allows N_3 to accept x. Thus N_3 accepts the language accepted by M^A. QED

Randomized algorithms

Consider a Turing Machine with the ability to flip coins (where each coin flip takes one unit of time).
Let $\mathbf{Prob}_M(x)$ denote the probability that a randomized Turing Machine M accepts x.
The following are some interesting classes that can be defined based on randomized Turing Machines.

Definition: A Language L is said to be in **PP**, iff there exists a polynomial time bounded randomized Turing Machine M such that:
(1) For all $x \in L$, $\mathbf{Prob}_M(x) > \frac{1}{2}$.
(2) For all $x \notin L$, $\mathbf{Prob}_M(x) < \frac{1}{2}$.

Definition: A Language L is defined to be in **BPP**, iff there exists a polynomial time bounded randomized Turing Machine M, and a constant $\epsilon > 0$, such that:
(1) For all $x \in L$, $\mathbf{Prob}_M(x) > \frac{1}{2} + \epsilon$.
(2) For all $x \notin L$, $\mathbf{Prob}_M(x) < \frac{1}{2} - \epsilon$.

Thus the probability of acceptance/rejection is bounded away from 1/2.

Definition: A Language L is defined to be in **R**, iff there exists a polynomial time bounded randomized Turing Machine M, such that:
(1) For all $x \in L$, $\mathbf{Prob}_M(x) > \frac{1}{2}$.
(2) For all $x \notin L$, $\mathbf{Prob}_M(x) = 0$.

Thus, the machine never makes an error on elements not in the language. For elements in the languages, it is correct at least 50% of the time. (Thus it has one-sided error)

For **ZPP** consider machines with three possible outputs. Yes, NO and ?.

Definition: A Language L is defined to be in **ZPP**, iff there exists a polynomial time bounded randomized Turing Machine M, with three outputs, Yes, No, and ?, such that:

(1.1) For all $x \in L$, $\mathbf{Prob}(M(x) = Yes) > \frac{1}{2}$.

(1.2) For all $x \in L$, $\mathbf{Prob}(M(x) = No) = 0$.

(2.1) For all $x \notin L$, $\mathbf{Prob}(M(x) = No) > \frac{1}{2}$.

(2.2) For all $x \notin L$, $\mathbf{Prob}(M(x) = Yes) = 0$.

Thus, for **ZPP** identification, machine commits no errors, and gives the right answer with at least 50% probability.

Testing whether a multi-variate polynomial is identically zero

Suppose we are given a polynomial $P(x_1, x_2, \ldots, x_m)$, in m variables, where the degree of each variable in the polynomial is at most d. The question is, is the polynomial identically zero?
Let us consider a randomized algorithm for the above problem.
Lemma: Suppose $P(x_1, x_2, \ldots, x_m)$, is a polynomial in m variables, where each variable has degree at most d. Let $M > 0$, be a positive integer.
If P is not identically zero, then the number of m-tuples $(x_1, x_2, \ldots, x_m) \in \{0, 1, \ldots, M - 1\}^m$, such that $P(x_1, \ldots, x_m) = 0$, is at most mdM^{m-1}.
Proof By induction on m.
Base Case: For $m = 1$, the lemma says that a uni-variate polynomial of degree $\leq d$, can have no more that d roots. This is standard result.

Induction Case:
Suppose the lemma holds for $m = n$.
That is, if a polynomial $Q(x_1, x_2, \ldots, x_n)$ is not identically zero, then the number of n-tuples $(x_1, x_2, \ldots, x_n) \in \{0, 1, \ldots, M-1\}^n$, such that $Q(x_1, \ldots, x_n) = 0$, is at most ndM^{n-1}.
We will show that the lemma holds for $m = n + 1$. Consider $P(x_1, \ldots, x_{n+1})$.
Write the polynomial as a polynomial in x_{n+1}, where the coefficients are polynomials in $x_1, \ldots, x_n$.
That is,
$P(x_1, \ldots, x_{n+1}) = P_d(x_1, \ldots, x_n) * x_{n+1}^d + P_{d-1}(x_1, \ldots, x_n) * x_{n+1}^{d-1} + \ldots + P_0(x_1, \ldots, x_n) * x_{n+1}^0$,
where P_i's are degree d polynomials in $x_1, \ldots, x_n$.

Now if P is 0 on some $(x_1, \ldots, x_{n+1})$, there are two possibilities:

(Case 1) Highest degree coefficient of x_{n+1} evaluates to 0, or

(Case 2) Highest degree coefficient of x_{n+1} does not evaluate to 0, but $P(x_1, \ldots, x_{n+1})$ is 0.
Now we analyse how many different values of $(x_1, \ldots, x_{n+1})$ can lead to above cases.
Case 1: Note that highest degree coefficient of x_{n+1} cannot be identically zero (since P is not identically zero).
Thus, by induction hypothesis, Case 1 can happen for at most ndM^{n-1} different values of $x_1, \ldots, x_n$.
Since, x_{n+1} itself can have at most M different values, we have that Case 1 can happen for at most ndM^n different values of $(x_1, x_2, \ldots, x_{n+1})$.

Case 2: For each different value of $x_1, \dots, x_n$, case 2 can happen for at most d different values of x_{n+1}.
Thus, case 2 can hold for at most dM^n different values of $(x_1, \dots, x_{n+1})$.
Summing up, we get that $P(x_1, \dots, x_{n+1})$ can be zero for at most $(n+1)dM^n$ different values of the tuple $(x_1, x_2, \dots, x_{n+1})$. QED

The lemma gives us a (randomized) method to test whether a polynomial $P(x_1, x_2, \dots, x_m)$, where each variable has degree at most d, is not identically zero.

```
TestNonZero:
Input: A polynomial $P(x_1, x_2, \ldots, x_m)$ with degree at most $d$ in
    each of the variables.
1. Let $M > 2md$.
2. Pick a random tuple $(x_1, x_2, \ldots, x_m) \in \{0, 1, \ldots, M-1\}^m$.
3. If $P(x_1, \ldots, x_m) \neq 0$, then output "Not identicaly zero".
4. Otherwise output "Probably identically zero".
End
```

It is easy to verify:
(1) If P is identically zero, then the above algorithm always outputs "Probably identically zero"
(2) If P is not identically zero, then the above algorithm outputs "Not identically zero" with probability at least $\frac{1}{2}$.
The run time of the above algorithm is bounded by a polynomial in m, d.

2-SAT

It can be shown that 2-SAT is in **P**.
However, the technique used in the following algorithm is useful in several other problems, and thus we will study it.

```
Input: (1) A set of $n$ variables.
(2) A set of $m$ clauses (with at most two literals per clause).
1. Initialize each variable to a random truth value.
2. For $i = 1$ to $N$, do
        If all the clauses are satisfied, then output "satisfiable" and
          stop.
        Otherwise pick the first unsatisfied clause. Randomly flip the
          truth value of one of the variables involved in the clause.
   EndFor
3. Output "Probably not satisfiable"
End
```

Note that, if the clauses are not satisfiable, then the algorithm always outputs, “Probably not satisfiable”.
We now argue that if the clauses are satisfiable, then with probability at least $\frac{1}{2}$, the algorithm outputs “satisfiable”, when we chose $N = 2n^2$.
Suppose A is one of the satisfying assignments for the clauses.
For ease of writing the proof, suppose $N = \infty$.
For any truth assignment A' let $dist(A, A')$ denote the number of variables on which the truth assignments A and A' differ.
$E(k)$: the expected number of flips to the variable assignment, when one starts the for loop with an A' such that $k = dist(A, A')$, before the algorithm above halts, i.e. reaches a satisfying assignment. (Here, A' is chosen in a worst case manner among all truth assignments which are at a distance k of A).

In 1 iteration of the For loop, the distance to A either becomes 1 smaller or 1 greater, where it becomes 1 smaller with probability at least half, and becomes 1 larger with probability at most half. Thus,

$$E(k) \leq \frac{E(k+1)+E(k-1)}{2} + 1.$$
$$E(0) = 0 \text{ and } E(n) = E(n-1) + 1.$$

(Where, we use $\leq$ since the algorithm may find a satisfying assignment different from A, or switching of both literals takes A' closer to A).

Note that if we replace $\leq$ in the above equation by $=$, to get equations:

$$E'(k) = \frac{E'(k+1)+E'(k-1)}{2} + 1.$$
$$E'(0) = 0 \text{ and } E'(n) = E'(n-1) + 1.$$

and then solve for E', then $E(k) \leq E'(k)$.

It can be verified that $E'(k) = 2kn - k^2$, satisfies the above equations.
Thus we have $E(k) \leq 2kn - k^2$, or $E(k) \leq n^2$.
Thus, whatever the initial value of truth assignments we start with, the expected number of iterations of the For loop before we reach A (or some other satisfying assignment) is at most n^2.
Thus with probability at least $1/2$, we take at most $2n^2$ iterations.

To see that $E(k) \leq E'(k)$, let $\Delta(k) = E'(k) - E(k)$.
As $\Delta(k) \geq \frac{\Delta(k+1)+\Delta(k-1)}{2}$, we have that $\Delta(k) \leq \Delta(k+1)$ implies $\Delta(k-1) \leq \Delta(k)$.
It thus follows from $\Delta(n-1) = \Delta(n)$ that for all $m < n$, $\Delta(m) \leq \Delta(m+1)$.
Now from $\Delta(0) = 0$, it follows that $\Delta(k) \geq 0$, for all $k \leq n$.

Boosting Probabilities

Lemma (Chernoff Bounds) Suppose we have a coin, which has probability p of coming up heads and probability $(1-p)$ of coming up tails.
Suppose we toss the coin n times.
Let X_n be the random variable denoting the number of heads (in n tosses).
Then $\mathbf{Prob}(X_n \geq (1+\theta)pn) \leq e^{-f(\theta)pn}$. For $\theta < 1$, one can take $f(\theta)$ to be $\theta^2/3$.

Proof: Note that $\mathbf{Prob}(e^{tX_n} \geq k\mathrm{E}(e^{tX_n})) \leq \frac{1}{k}$. Thus,

$\mathbf{Prob}(X_n \geq (1+\theta)pn) = \mathbf{Prob}(e^{tX_n} \geq e^{t(1+\theta)pn}) \leq e^{-t(1+\theta)pn}\mathrm{E}(e^{tX_n})$

(by choosing $k = \frac{e^{t(1+\theta)pn}}{\mathrm{E}(e^{tX_n})}$).
Now, $\mathrm{E}(e^{tX_n}) = (\mathrm{E}(e^{tX_1}))^n = [(1-p)+pe^t]^n$.

Thus,

$$\mathbf{Prob}(X_n \geq (1+\theta)pn) \leq e^{-t(1+\theta)pn}[1+p(e^t-1)]^n$$

Thus, since $(1+a)^n \leq e^{an}$,

$$\mathbf{Prob}(X_n \geq (1+\theta)pn) \leq e^{-t(1+\theta)pn}[e^{pn(e^t-1)}]$$

Taking $t = \ln(1+\theta)$, we have

$$\mathbf{Prob}(X_n \geq (1+\theta)pn) \leq e^{pn(\theta-(1+\theta)ln(1+\theta))}$$

Thus one can take $f(\theta) = [(1+\theta)ln(1+\theta))] - \theta$.
For $\theta < 1$,
$\ln(1+\theta) = \theta - \frac{\theta^2}{2} + \frac{\theta^3}{3} - \ldots.$ Thus, $(\theta - (1+\theta)ln(1+\theta)) = \frac{-\theta^2}{2} + \frac{\theta^3}{6} - \frac{\theta^4}{12} \cdots \leq -\frac{\theta^2}{3}$. QED

Chernoff Bounds allows us to boost the correctness probability to quite close to 1, in **BPP**, **R** and similar classes.

Theorem: Suppose $L \in \mathbf{BPP}$. Then there exists a polynomial time bounded probabilistic Turing Machine M such that
(1) $x \in L \Rightarrow \mathbf{Prob}_M(x) \geq 1 - 2^{-|x|}$.
(2) $x \notin L \Rightarrow \mathbf{Prob}_M(x) \leq 2^{-|x|}$.

Proof Suppose M is such that
(1) $x \in L \Rightarrow \mathbf{Prob}_M(x) \geq \frac{1}{2} + \epsilon$.
(2) $x \notin L \Rightarrow \mathbf{Prob}_M(x) \leq \frac{1}{2} - \epsilon$.
Let $p =$ maximum probability of error $= \frac{1}{2} - \epsilon$.
Let $\theta = \epsilon$.

Consider the following algorithm M'

```
$M'(x)$
1. Let $n = \frac{6|x|}{\epsilon^2 - 2\epsilon^3}$.
2. Run $M(x)$, $n$ times.
3. Output the answer, which is obtained in the majority of the above
      runs.
End
```

Note that the probability of error in each run is at most $p = \frac{1}{2} - \epsilon$.
Let $\theta = \epsilon$.
Thus, the probability that in n trials, the number of wrong answers is more than $(\frac{1}{2} - \epsilon)(1 + \epsilon)n$ (which is $< \frac{n}{2}$) is at most $e^{-|x|} \leq 2^{-|x|}$ (where the first inequality is by Chernoff Bounds). QED

Note that

$\mathbf{ZPP} \subseteq \mathbf{R} \subseteq \mathbf{BPP} \subseteq \mathbf{PP}$.

$\mathbf{ZPP} \subseteq \mathbf{R} \subseteq \mathbf{NP} \subseteq \mathbf{PP}$.

Theorem: $\mathbf{BPP} \subseteq \Sigma_2^p$.
Proof: Suppose $L \in \mathbf{BPP}$. Suppose M is a machine such that
(1) if $x \in L$, then M accepts L with probability $> 1 - 2^{-|x|}$.
(2) if $x \notin L$, then M accepts L with probability $< 2^{-|x|}$.
Suppose M is $p(|x|)$ time bounded on input x, where p is a polynomial.
Let $A_x = \{y : |y| = p(|x|)\}$.
Let, $B_x = \{y : |y| = p(|x|) \text{ and } M(x, y) \text{ accepts}\}$.
(Here, when we say $M(x, y)$ accepts we mean $M(x)$ accepts, when the random tosses are according to y.)
Let $C_x = A_x - B_x$.
Intuitively, B_x is tosses, on which M accepts, and C_x are the tosses on which M rejects.

Thus, we have
(1) if $x \in L$, then $card(B_x)/card(A_x) > 1 - 2^{-|x|}$.
(2) if $x \notin L$, then $card(B_x)/card(A_x) < 2^{-|x|}$.
Let $k = \frac{p(|x|)}{|x|}$.
Let $y \oplus z$ denote bitwise xor. Now, for each x, consider the following set:
$D_x = \{(y_1, y_2, \ldots, y_k) : y_i \in A_x$ and $(\forall z \in A_x)(\exists i : 1 \leq i \leq k)[y_i \oplus z \in B_x]\}$.

Claim: For large enough x, if $x \in L$, then D_x is not empty.
Proof: For each z, let $S_z = \{(y_1, \ldots, y_k) : (\forall i : 1 \leq i \leq k)[y_i \in A_x$ and $y_i \oplus z \in C_x]\}$.
Intutively, S_z are bad tuples.
Note that $card(S_z) \leq [card(C_x)]^k$.
Thus, $card(\cup_{z \in A_x} S_z) < 2^{p(|x|)} * 2^{k(p(|x|)-|x|)} = 2^{p(|x|)+k(p(|x|)-|x|)} = 2^{kp(|x|)}$.
Thus, for large enough x, there exists a $(y_1, y_2, \ldots, y_k)$ such that $(y_1, y_2, \ldots, y_k) \notin \cup_{z \in A_x} S_z$; i.e., $(\forall z \in A_x)(\exists i : 1 \leq i \leq k)[y_i \oplus z \notin C_x]$. Thus, $(y_1, y_2, \ldots, y_k) \in D_x$.

Claim: For large enough x, if $x \notin L$, then D_x is empty.
Proof: Suppose $x \notin L$.
If D_x, is not empty, then let $(y_1, \ldots, y_k) \in D_x$.
Thus, for each $z \in A_x$, there exists a y_i, such that $M(x, y_i \oplus z)$ Accepts.
But this implies (by pigeonhole principle), that B_x contains at least $2^{p(|x|)}/k$ elements.
A contradiction for large enough x.
QED Claim

From the above claims it follows that
For large enough x,
$(x \in L) \Leftrightarrow (\exists (y_1, \ldots, y_k) : y_i \in A_x)(\forall z \in A_x)[(\exists i : 1 \leq i \leq k)[y_i \oplus z \in B_x]]$.
But, the above is a Σ_2^p formula.
Thus $L \in \Sigma_2^p$.
QED Theorem

Note that $\Sigma_2^p = NP^{NP}$. Thus, $BPP \subseteq NP^{NP}$.