More Classes with Randomness.

CS254
Chris Pollett
Nov. 6, 2006.

Outline

- Randomized Algorithms
- Randomized Complexity classes

Random Walks for SAT

- Last day, we presented the following randomized algorithm forf SAT:

1. Start with any truth assignment T, and repeat the following r times:
   - If there is no unsatisfied clause output “Satisfiable”, halt.
   - Otherwise, take any unsatisfied clause; pick any of its literals at random and flip its value
2. After r repetitions reply “the formula is probably unsatisfiable”

- We had not yet determined what is a reasonable r to use when running the algorithm

Theorem

Suppose that the random walk algorithm with $r=2n^2$ is applied to any satisfiable instance of 2SAT with n variables. Then the probability that a satisfying truth assignment will be discovered is at least 1/2.

Proof. Let T be a truth assignment which satisfies the given 2SAT instance I. Let t(i) denote the number of expected repetitions of the flip step until a satisfying assignment is found starting from anassignment T´ which differs in at most i positions from T. Notice:

1. $t(0) = 0$
2. If we find some other satisfying assignment we do not need to continue
3. Otherwise, we flip at least once, and we have a 50% chance of moving closer to the solution; 50% farther. So $t(i) \leq 1/2(t(i-1) + t(i+1))+1$
4. We also have $t(n) \leq t(n-1) + 1$(If every literal is wrong, we can only move closer).

The worst case is the when relation t of 3 holds as an equation. $x(0)=0$; $x(n)=x(n-1)+1$; $x(i) = 1/2(x(i-1)+x(i+1))+1$

Proof Continued

$$
\begin{array}{rcl}
x(1) & = & 1/2[x(0)+x(1)]+1 \\
\vdots & = & \vdots \\
x(n-2) & = & 1/2[x(n-3)+x(n-1)]+1 \\
x(n-1) & = & 1/2[x(n-2)+x(n)]+1 \\
+ \quad x(n) & = & x(n-1)+1 \\
\hline
x(n)+(\sum_{i=2}^{n-1} x_i)+x(1) & = & 1/2x(n)+1/2x(n-1)+(\sum_{i=2}^{n-1} x_i)+1/2x(1)+n \\
x(n)+x(1) & = & 1/2x(n)+1/2x(n-1)+1/2x(1)+n \\
x(n)+x(1) & = & 1/2x(n)+1/2(x(n)-1)+1/2x(1)+n \\
1/2x(1) & = & n-1/2
\end{array}
$$

Adding all the x(i)'s together gives: $x(1) = 2n-1$.

Then solving the x_1 equation for x_2 gives 4n-4, and in general, $x(i) = 2in-i^2$.

Thus we have shown $t(i) \le x(i) \le x(n) = n^2$. Now consider the following lemma:

Lemma (Markov Inequality). If x is a random variable taking nonnegative integer values, then for any $k>0$, $prob[x \ge k*E(x)] \le 1/k$.

Proof. Let p_i be the probability that x=i.

$E(x) = \sum_i i*p_i = \sum_{i \le k*E(x)} i*p_i + \sum_{i > k*E(x)} i*p_i > k*E(x)*prob[x>k*E(x)]$ Q.E.D.

The theorem then follows taking k=2.

Randomized Complexity Classes

- We would like to study randomized algorithms formally in the context of Turing Machines.
- To do this we will need to consider a variation of what it means for a nondeterministic machine accepts:

Defn. Let N be a p-time NDTM. Assume N is also precise (halts in the same number of steps for all inputs of a given length) and always has exactly two nondeterministic moves from any position. Let L be a language. N is a p-time *Monte Carlo algorithm* for L, if whenever x is in L at least half of N's branches on x accept. Further, if x is not in L then all computation paths halt no. The class of all p-time Monte Carlo languages is denoted **RP.**

RP is robust

- Suppose we used some value $e < 1/2$ rather than 1/2 in the definition of RP.
- Let L be a language in RP which is accepted according to some machine M where at least e fraction of the branches must accept.
- We could repeat M execution k-times and report yes if any of these executions reported yes. The chance of issuing a false negative is then $(1-e)^k$.
- Take $k = [-1/\log(1-e)]$ makes the probability if x is in the language at least 1/2.

Semantic versus Syntactic Complexity Classes

- There is no easy way to tell if a given TM N satisfies the Monte Carlo condition.
- For instance, for NP the sequence of guesses leading down an accepting path is a certificate of being in the language. If no such certificate exists then we’re not in the language.
- For RP, having at least 1/2 the paths accept says we’re in the language, but the absence of 1/2 the paths accepting is not the condition for not being in the language -- If we’re not in the language we must have all paths rejecting.
- This is similar to the situation for NP∩coNP and TFNP.
- These classes are called semantic classes in contrast to classes like P and NP which are called syntactic classes.
- Semantic classes don’t usually have complete problems.

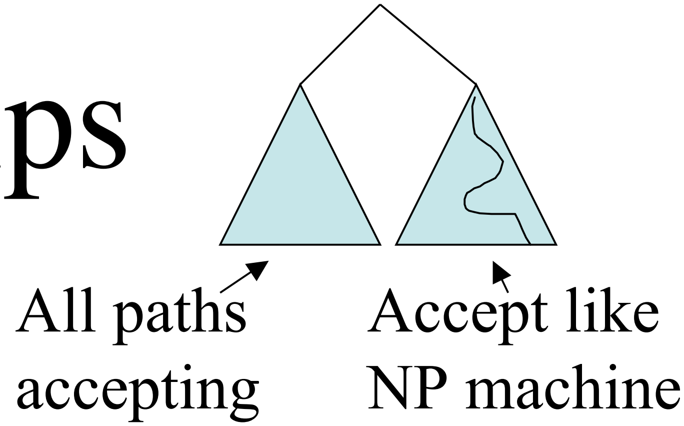
RP and other Classes

- RP is contained in NP. (A machine recognizing L in RP also recognizes that L is in NP).
- It is unknown if RP is contained in NP $\cap$coNP
- Given the asymmetry between acceptance and rejection in RP, it is natural to consider the class coRP.
- We define ZPP = RP $\cap$coRP. This is sometimes called the class of languages with Las Vegas algorithms
- It used to be Primes was only known to be in ZPP. Now its known to be in P.

PP

- Consider the problem MAJSAT: Given a boolean expression, do a majority of the assignments satisfy it?
- There is an obvious certificate for this problem namely on n variables give $2^{n-1}+1$ satisfying truth assignments.
- We let PP be the class of languages L which are recognized by precise NTMs with two branches at every step, such that when x is in L more than half the branches accept and when x is not in L at most half the branches accept.
- You can show MAJSAT is PP-complete under logspace reductions.

Relationships



Theorem NP⊆ PP.

Proof. Suppose L is in NP by machine N. Let N´ be identical to N except that it has a new initial state with two nondeterministic choices out of it. On the first branch, we run for the same number of steps as N (always branching each step two ways) and along every path we accept. On the second branch we simulate N. So N´ will have more than half its paths accepting iff N has at least one accepting path.