CS5120: Probability & Computing	**Jan-May 2020**
Lecture 16/17	
The Method of Conditional Expectations	
Lecturer: N.R.Aravind	*Scribe: N.R.Aravind*

1 3-SAT and random assignments

A k-SAT instance φ is a set of clauses, where each clause is an OR of k distinct literals. An instance in which every clause has at most k distinct literals will be called a partial k-SAT instance; a partial instance can have empty clauses which are assumed to be True. We denote by m the number of clauses, and n the number of variables.

An example of a 3-SAT instance is $\varphi = \{(x \vee \bar{y} \vee \bar{z}), (\bar{x} \vee z \vee \bar{w})\}$ with $m = 2$ clauses, over $n = 4$ variables. An example of a partial 3-SAT instance is $\varphi(x = F) = \{(\bar{y} \vee \bar{z}), (T)\}$.

We denote by $\mu(\varphi)$, the expected number of clauses satisfied by a random assignment to the variables of φ.

Observation:

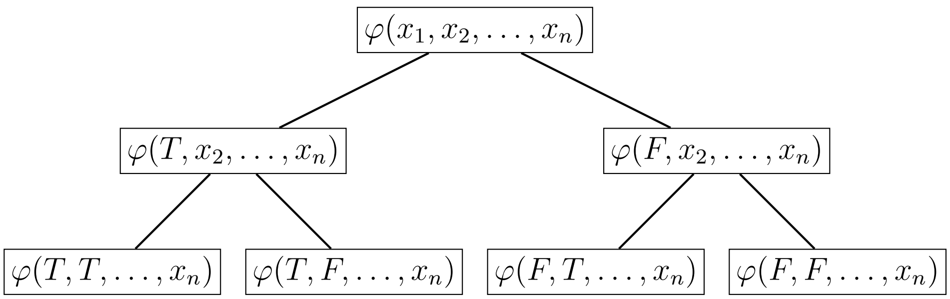
- If φ is an instance of 3-SAT, then the probability of each clause being satisfied is $1 - 1/8 = 7/8$; thus $\mu(\varphi) = 7m/8$.
- If φ is a partial 3-SAT instance, with m_0, m_1, m_2, m_3 denoting the number of clauses of size $0, 1, 2, 3$ respectively, then $\mu(\varphi) = m_0 + \frac{1}{2}m_1 + \frac{3}{4}m_2 + \frac{7}{8}m_3$.

2 Finding an assignment satisfying many clauses

For a 3-SAT instance φ with m clauses, we saw that the expected number of clauses satisfied by a random assignment is $7m/8$. Can we actually find an assignment satisfying these many clauses? That's the goal of this section.

Let φ be a 3-SAT instance with n variables and m clauses. Consider a binary tree whose root is φ, with leaves being the 2^n possible assignments, and where each vertex at the ith level (for $i = 1, 2, \ldots, n$) has two children, one corresponding to $x_i = T$ and the other corresponding to $x_i = F$.

The first two levels of this tree are illustrated below. A vertex at depth i corresponds to a partial 3-SAT instance, in which the first i variables have been assigned values.



Our algorithm will successively choose a truth-value for each of $x_1, \ldots, x_n$. Which value should we choose for x_1? Equivalently, which of the two subtrees ($\varphi|x_1 = T$) vs $\varphi|x_1 = F$) should we choose?

The idea is that we choose the "heavier" subtree, where the "weight" of a subtree rooted at node v is equal to $\mu(\varphi_v)$; here φ_v is the partial 3-SAT instance at node v.

Thus, the algorithm is the following:

- Set u to be the root.
- For $i = 1$ to n, do:
- If $\mu(\varphi_u|x_i = T) > \mu(\varphi_u|x_i = F)$, then set $x_i = T$ and u to be the child node(of current u) corresponding to $x_i = T$. Else set $x_i = F$ and u to be the child node corresponding to $x_i = F$.

We claim that after every iteration, $\mu(\varphi_u)$ stays the same or increases (that is, is non-decreasing). Thus, when the algorithm reaches a leaf node, corresponding to an assignment $x_1 = a_1, \ldots, x_n = a_n$, where each $a_i \in \{T, F\}$, the number of clauses satisfied by this assignment is at least as large as $\mu(\phi) = \dfrac{7m}{8}$.

To prove the claim, let u be a node with two children v, w, where v corresponds to $x_i = T$ and w to $x_i = F$. Then note that:

$$\mu(\varphi_u) = \frac{1}{2}\mu(\varphi_u | x_i = T) + \frac{1}{2}\mu(\varphi_u | x_i = F) = \frac{1}{2}\left(\mu(\varphi_v) + \mu(\varphi_w)\right).$$

Thus, $max\left(\mu(\varphi_v), \mu(\varphi_w)\right) \geq \mu(\varphi_u)$, which proves the claim, and completes the argument for correctness of the algorithm.

Analysis of running time: A key point of the above algorithm is that computing φ_u for a node u can be done efficiently (in polynomial time), since from section 1, it is a linear combination of the number of clauses with 0,1,2,3 literals; these numbers can be counted in $O(m)$ time. Thus each iteration takes $O(m)$ time, so that the total running time is $O(mn)$.