Lecture - 38

Thursday, 10 Nov, 17:10- 18:00

2 SAT- Markov Chains

In this lecture, we see how we can use the Markov Chains for the analysis of a randomised algorithm for solving 2-SAT problem.

1 2 SAT

You might be knowing the satisfiability problem, commonly known as SAT problem. Given a boolean formula in the CNF(Conjunctive Normal Form) i.e. as conjunctions(AND) of disjunctions(OR), can you find an assignment to the variables such that the formula is True? An example is shown below.

$$(x1 \vee x2 \vee x3') \wedge (x1' \vee x3) \wedge (x1' \vee x2' \vee x3') \wedge (x1 \vee x2')$$

Here, $x1, x2, ..$ are called the variables and $x1, x1', x2, x2', ...$ are called literals. So now can you give an assignment to the variables $x_1, x_2, ..$ such that the formula is True.

The general SAT formula is NP Hard. A SAT formula with the restriction that every clause has exactly k literals is called k-SAT. A SAT formula where every clause has exactly 2 literals is called $2-$SAT. $2-$SAT is polynomial time solvable.

We see a randomised way of solving the $2-$SAT and look at its analysis.

2 Randomised Algorithm and Markov Chain

The algorithm is very simple. We start with assigning random values to all the variables and see if the formula is getting satisfied. If yes, we are done. If no, then there is atleast one clause which is not getting satisfied. Let this clause be $C = X \vee Y$, where X and Y are two literals. C is not satisfied, this means both the literals are false and flipping the value of any of the variables associated with these literals will satisfy this clause. So, whenever the current assignment is not satisfying the formula, we pick an unsatisfied clause and flip the value of one of its variable picked uniformly at random.

2.1 Where is Markov Chain in it?

We now represent this system in the form of a Markov Chain. Before doing that, please see that there can be many satisfying assignments for this formula. We pick on one of these assignments and call it S. Assume there are n variables and m clauses. Our algorithm will be producing different assignments to the variables till it reaches satisfying assignment. We associate a number with every assignment say A_i. S has come assignments. A_i has some assignment. The number X_i = The number of variables which have the same assignment in S as in A_i. Please note that there are 2^n total possible ways of assignments. Also note that there can be more than one possible assignments A_i and A_j such that $X_i = X_j$.

We consider a Markov Chain having $n+1$ states numbered $s_0, s_1, s_2, ..., s_{n-1}, s_n$. The process is said to be in state s_j if for the current assignment $A_i, X_i = j$, i.e. there are j variables which agree with their assignments in S. The system will move from one state to other. The system succeeds when it reaches A_k such that $X_k = n$. Please note that the chain can get over before this state as well, if it reaches some other satisfying assignment other than S. But we are interested in finding the time it will take to reach the satisfying assignment, which will only increase in case we consider just S. So, what we will be getting is an upper bound. This has been shown in Figure 1.

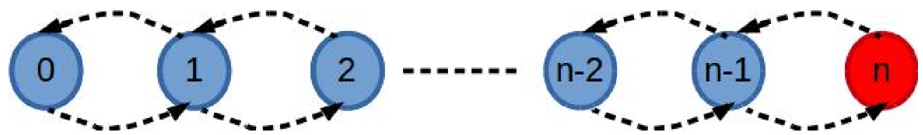

Fig. 1: Markov Chain for 2SAT

Since, we can change the value of one variable in one step, we can jump either one state towards the left or one state towards the right, not more than that. We now have a Markov chain. What is missing - The probabilities of jumping from one state to the other, i.e. the transition matrix. Let us represent the transition matrix for this Markov Chain as P, where P_{ij} represent the probability of jumping from s_i to s_j. Please note that $P_{ii} = 0$. Also $P_{ij} = 0,\ if\ |i-j| \geq 2$.

1. s_0 has only one outgoing path to s_1. Hence, $P_{01} = 1$.
2. If $1 \leq i \geq n-1$, then let us see what will happen? When we flip the value of a variable, if the flipped value matches its value in S, we move one state right, else we move one state left.
   We have now encountered an unsatisfied clause having 2 literals. Assume that S has values of these variables as $\{\alpha, \beta\},\ where\ \alpha, \beta \in \{0, 1\}$. Our clause before flipping can have one of these three pair of values : $\{1-\alpha, 1-\beta\}, \{\alpha, 1-\beta\}, \{1-\alpha, \beta\}$. If it was the fourth possible assignment, this clause would have been satisfied.
   - Case 1:$\{1-\alpha, 1-\beta\}$: Flip any of the variables, the number of variables matching values in S will increase by 1. $P_{i,i+1} = 1$
   - Case 2:$\{\alpha, 1-\beta\}$: Flip β, the number of variables matching values in S will increase by 1. $P_{i,i+1} = 1/2$
   - Case 3:$\{1-\alpha, \beta\}$: Flip α, the number of variables matching values in S will increase by 1. $P_{i,i+1} = 1/2$

   Hence, $P_{i,i+1} \geq 1/2$

This has been shown in Figure 2.

We are interested in knowing the time it will take for the system to reach the red state. It is difficult to analyse this Markov Chain. So, we take a stricter version of this chain. This has been shown in

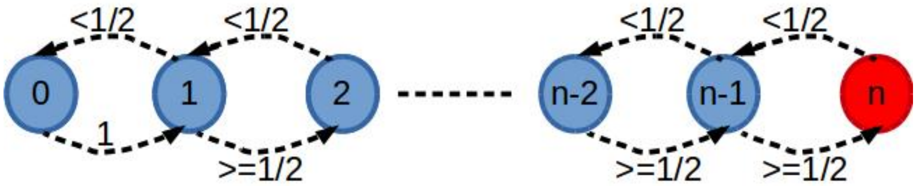

Fig. 2: Markov Chain for 2SAT

Figure 3. Please note that the time taken in this chain will be higher than the previous version. Hence, we will be obtaining an upper bound on the time and show that it is actually less.

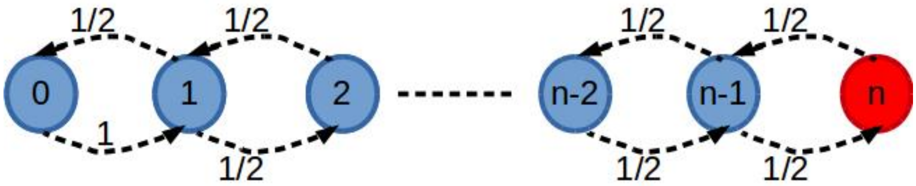

Fig. 3: Markov Chain for 2SAT- Stricter Version

So, now we can depict our Markov chain with the following set of equations

$$s_{i+1} = \begin{cases} s_1 & \text{if } s_i = 0 \\ s_i + 1 & \text{with probability} \frac{1}{2} \\ s_i - 1 & \text{with probability} \frac{1}{2} \end{cases}$$

3 Time Taken to Find Satisfying Assignment

Let h_j represent the expected number of steps by the algorithm to reach s_n when the initial configuration is at s_j.

$h_1 = h_0 + 1$, since $P_{01} = 1$
$h_n = 0$

Let us now look at the expected number of steps it takes to reach from s_j to s_n where, $1 \leq j \leq n-1$.

We know that the $p_{j,j+1} = \frac{1}{2}$, if $1 \leq j \leq n-1$

Also, $p_{j,j-1} = \frac{1}{2}$, if $1 \leq j \leq n-1$

The system arrives at s_j with probabilities $\frac{1}{2}$ if it was at one of the states s_{j-1} or s_{j+1} in the previous states.

Let Z_j be the number of steps required from step j.

Then, $E[Z_j] = \frac{1}{2}(1 + E[Z_{j+1}]) + \frac{1}{2}(1 + E[Z_{j-1}])$
or
$h_j = \frac{1}{2}(1 + h_{j+1}) + \frac{1}{2}(1 + h_{j-1})$

We now have a system of recurrence relations to solve :

1. $h_0 = h_1 + 1$
2. $h_n = 0$
3. $h_j = \frac{1}{2}(1 + h_{j+1}) + \frac{1}{2}(1 + h_{j-1})$, $1 \leq j \leq n - 1$

One way to solve this recurrence relation is to use the characteristic equation of linear homogenous recurrence relations with constant coefficients, but we try to solve this recurrence from the first principles.

We try to modify the third recurrence and express h_j in terms of only h_{j+1} by eliminating h_{j-1} from the expression.

$h_1 = 1 + h_0/2 + h_2/2$ (From third point)
$h_1 = 1 + 1/2 + h_1/2 + h_2/2$(From First Point)
$h_1/2 = 1 + 1/2 + h_2/2$.....(eq 1)

Similarly, $h_2 = 1 + h_1/2 + h_3/2$ (From third point)
$h_2 = 1 + 1 + 1/2 + h_2/2 + h_3/2$(From eq 1)
$h_2/2 = 1 + 1 + 1/2 + h_3/2$.....(eq2)

Similarly, one can see that $h_j/2 = j + 1/2 + h_{j+1}/2$ for $1 \leq j \leq n - 1$
or $h_j = 2j + 1 + h_{j+1}$

So, we now can write three equations as.

One way to solve this system is to use the direct solution of the linear homogenous recurrence rel

1. $h_0 = h_1 + 1$
2. $h_n = 0$
3. $h_j = 2j + 1 + h_{j+1}$, $1 \leq j \leq n - 1$

Now, we find h_0
$h_0 = h_1 + 1$
$= 2 + 1 + h_2$

$= 2 + 1 + 4 + 1 + h_3$
$= 2 + 1 + 4 + 1 + 6 + 1 + h_4$
$= 2 + 1 + 4 + 1 + 6 + 1 + ... + 2(n - 2) + 1 + 2(n - 1) + 1 + h_n$
$= 2 + 1 + 4 + 1 + 6 + 1 + ... + 2(n - 2) + 1 + 2(n - 1) + 1$, since $h_n = 0$

So, $h_0 = 2(1 + 2 + ... + n - 1) + n - 1$
$= 2 \times (n - 1)/2(2 + n - 2) + n - 1$
$= n(n - 1) + n - 1$
$= n^2 - 1$

Hence, it takes maximum of $O(n^2)$ steps to find a satisfying assignment.

4 Ransdomised 3 SAT

Can you now reproduce the above analysis for randomised 3-SAT and find its maximum expected running time. It turns out to be $O(2^n)$, which is very costly.

In the next lecture, we will use some observations to modify the algorithm and better the running time.