CS5120: Probability & Computing **Jan-May 2020**

Markov Chains 2

1 Expected time on a path

We consider the path graph on $V = \{v_0, v_1, \ldots, v_n\}$, with the edge-set being $\{v_0v_1, v_1v_2, \ldots, v_{n-1}v_n\}$. Suppose that we start a random walk at v_0. What is T_{0n}, the expected number of time-steps to reach v_n from v_0?

In the following, we answer this question as well as for the case when the walk is biased, that is: for every $i \geq 1$, the probability of moving from v_i to v_{i+1} is p and the probability of moving from v_i to v_{i-1} is $1-p$. Even in the general model, we will assume that the probability of moving from v_0 to v_1 is equal to one.

Proposition 1

1. *For $p = 1/2$, we have: $T_{0,n} = n^2$.*
2. *For $p > 1/2$, we have: $T_{0,n} = \Theta(n)$.*
3. *For $p < 1/2$, we have: $T_{0,n} = \Theta\left(c_p^n\right)$, where $c_p = \dfrac{1}{p} - 1$.*

Proof We consider the first case, that is: $p = 1/2$. For i, j, let T_{ij} denote the expected number of steps to reach from i to j, and we use the convention that $T_{ii} = 0$.

By calculation, we can find that $T_{01} = 1, T_{02} = 4, T_{03} = 9$; this leads to the guess that $T_{0n} = n^2$. We have:

$T_{i,i+1} = \dfrac{1}{2}(1) + \dfrac{1}{2}(1 + T_{i-1,i+1})$ and since $T_{i-1,i+1} = T_{i-1,i} + T_{i,i+1}$, we get: $T_{i,i+1} = 1 + \dfrac{1}{2}(T_{i-1,i} + T_{i,i+1})$. This simplifies to: $T_{i,i+1} = T_{i-1,i} + 2$. This recurrence along with $T_{0,1} = 1$ implies that $T_{i,i+1} = 2i + 1$ for $i \geq 0$. Finally, we have:

$$T_{0,n} = T_{0,1} + T_{1,2} + \ldots + T_{n-1,n} = \sum_{i=0}^{n-1}(2i+1) = \sum_{i=1}^{n}(2i-1) = n^2.$$

This completes part 1 of the proposition. The other two parts can be completed via the exercise below. ■

Exercise 1/HW 3: For a biased random walk, where $Pr[X_{t+1} = i + 1 | X(t) = i] = p$, show that

$$T_{i,i+1} = \frac{1}{p} + \left(\frac{1}{p} - 1\right) T_{i-1,i}.$$

Conclude that when $p < 1/2$, we have $T_{i,i+1} = \Theta\left(c_p^i\right)$, where $c_p = \frac{1}{p} - 1$, and thus deduce part 3 of Proposition 1. For $p > 1/2$, conclude that $T_{i,i+1} = \Theta(1)$ and thus $T_{0,n} = \Theta(n)$.

Exercise 2: Find $T_{i,j}$ for $i \leq j$ for the $p = 1/2$ case.

2 Randomized algorithm for 2-SAT

Recall that a k-SAT instance φ is a set of clauses, where each clause is an OR of k distinct literals. We assume that the variable set is $\{x_1, \ldots, x_n\}$ and we say φ is satisfiable if there is an assignment $(a_1, \ldots, a_n)$ of truth values to $x_1, \ldots, x_n$ such that every clause of φ is made True (or satisfied). We denote by m the number of clauses. The following algorithm makes sense for any instance of SAT. **RANDOM-WALK-SAT:**

1. Fix an arbitrary assignment $(x_1, \ldots, x_n) = (a_1, \ldots, a_n)$.
2. While there exists a false clause C, do:
3. Pick a random literal (uniformly) from C and flip the value of its variable. For example, if the clause is $\{x, \neg y\}$, with the current assignment being $x = F, y = T$, then with probability $1/2$, x is changed to T (and y unchanged), and with probability $1/2$, y is changed to F (x unchanged).
4. Re-evaluate the values of all clauses.
5. End while
6. Output the assignment.

Proposition 2 *If φ is a satisfiable instance of 2-SAT, then the expected time for RANDOM-WALK-SAT to reach a satisfying assignment is at most n^2.*

Proof Suppose that φ is satisfiable and let $b = (b_1, \ldots, b_n)$ be a satisying assignment. Note that the algorithm defines a Markov chain on the Hamming cube $\{0,1\}^n$, but we don't know the transition probabilities. What we want to estimate is the expected time to reach the assginment b from the initial assignment a (which was picked arbitrarily).

After t steps of the algorithm, let X_t be the number of variables in which the current assignment agrees with b. Note that when the algorithm flips a variable, it can lead to $X_{t+1} = X_t - 1$ or to $X_{t+1} = X_t + 1$. We thus want to calculate the expected time t for which X_t becomes equal to n (all variables agree with b). The value that each X_t can take are in $\{0, 1, \ldots, n\}$. Thus the algorithm performs a kind of random walk on $\{0, 1, \ldots, n\}$.

We claim that the probability of moving towards b is at least $1/2$. To see this, consider a false clause $C = \{l_1, l_2\}$, where l_1, l_2 are false under the current assignment of the algorithm. In the assignment under b, the value of (l_1, l_2) can be one of $(T,F), (F,T), (T,T)$. If this value were (T,F) and the algorithm flipped l_2 instead of l_1, then the assignment moves away from b, that is, if $X_t = i$, then $X_{t+1} = i - 1$. If the algorithm flipped l_1 instead of l_2, then the assignment moves towards b, that is: $X_{t+1} = i+1$. Thus, in the first two cases, with probability $1/2$, the algorithm moves the assignment towards b. In the third case, no matter which literal is flipped by the algorithm, the assignment is moved closer (by Hamming distance one) to b. This proves the claim in the beginning of this paragraph.

We have thus seen that $Pr(X_t = i + 1 | X_t = i) \geq 1/2$. By comparing this with the random walk Markov chain (where the probability is equal to $1/2$), we can conclude that the expected time until $X_t = n$ is at most n^2. This completes the proof of Proposition 2. ■

Note that RANDOM-WALK-SAT is a Las Vegas algorithm, and if φ is not satisfiable, then the algorithm doesn't terminate. Therefore we will forcefully terminate the While loop after N steps, and by Markov's inequality, the probability that it doesn't find a satisfying assignment (if φ is satisfiable) is at most $\dfrac{n^2}{N}$. A choice of $N = 10n^2$ would thus ensure success probability at least 0.9.

3 Analysis of RANDOM-WALK-SAT for 3-SAT

How well does RANDOM-WALK-SAT do an instance of SAT in which clauses can have 3 or more literals?

Consider a clause $C = \{l_1, l_2, l_3\}$ that is false under the current assignment. The algorithm flips a random literal. What can we say about the probability that this flip moves the assignment towards the satisfying assignment? If the satisfying assignment has exactly one true literal in C, then the probability that the algorithm flips this literal is $1/3$.

In general, if the clause has k literals and the satisfying assignment makes$1 \leq l \leq k$ literals true, then the probability that the algorithm moves towards the assignment is $\frac{l}{k}$. Because we don't know the value of l, we can only conclude that this probability is at least $\frac{1}{k}$.

Thus, for $k = 3$, by applying Proposition 1, the expected time to find a satisfying assignment can only be upper-bounded by $O(2^n)$, which doesn't help, as the trivial algorithm of checking all possible assignments runs in this time.

4 Schöning's algorithm for 3-SAT:

In 1987, Monien & Speckenmeyer gave the first non-trivial algorithm for 3-SAT (over n variables), running in time $O^*(1.61^n)$. This was improved to $O^*(1.38^n)$ in 1998 by Paturi, Pudlak, Saks, Zane (PPSZ); in 2011, the same algorithm was proved to run in $O^*(1.308^n)$ time by Hertli. This algorithm is also randomized like the one we will see, but the analysis is quite intricate.

In 1999, Schöning gave the following simple randomized algorithm for 3-SAT, which runs in expected time $O^*(1.33^n)$.

RANDOM-WALK-SAT with RESTART:

1. Pick a **random** assignment $(x_1, \ldots, x_n) = (a_1, \ldots, a_n)$.
2. Repeat up to $3n$ times, terminating if a satisfying assignment is found.
    - If there exists a false clause C, do:

- Pick a random literal (uniformly) from C and flip the value of its variable.
- Re-evaluate the values of all clauses.

3. If the assignment does not satisfy all clauses, GOTO Step 1 (restart).
4. Output the assignment.

One of the obvious ideas behind the restart is that of probability amplification, but there are two other key ideas.

1. Terminating the algorithm if it doesn't succeed soon is like cutting our losses and preventing the assignment from going even further away from the satisfying assignment.

2. When we pick a random assignment, there is a tiny probability that it is very close to the satisfying assignment, and over many random assignments, the initial distance from the satisfying assignment can be expected to be close a few times, which improves the overall success probability.

Claim 1 *Consider a random walk on* $\{0, 1, \ldots, n\}$*, where* $Pr[X_{t+1} = i + 1 | X_t = i] = 1/3$ *and* $Pr[X_{t+1} = i - 1] = 2/3$*. Let* q_j *be the probability of reaching* n *within* $3n$ *steps, when the initial vertex is* $n - j$*. Then for* $j > 0$*, we have:* $q_j \geq \dfrac{c}{\sqrt{j}2^j}$ *for some constant* $c > 0$*.*

Assuming this claim, the probability that the algorithm finds the satisfying assignment b within $3n$ steps starting from a random assignment a is at least:

$$\frac{1}{2^n} + \sum_{j=1}^{n} Pr\left(H(a,b) = j\right) \frac{c}{\sqrt{j}2^j} \geq \frac{c}{\sqrt{n}} \sum_{j=0}^{n} \binom{n}{j} \frac{1}{2^n} \frac{1}{2^j},$$

which is equal to $\dfrac{c}{\sqrt{n}2^n} \left(1 + \dfrac{1}{2}\right)^n = \dfrac{c}{\sqrt{n}} \left(\dfrac{3}{4}\right)^n$.

Thus, the expected number of Restarts needed by the algorithm is $O^*\left((4/3)^n\right)$. If the number of restarts is fixed in advance to $r(4/3)^n \dfrac{\sqrt{n}}{c}$, then the probability that the algorithm does not find a satisfying assignment when φ is satisfiable, is at most $\dfrac{1}{2^r}$.

We now prove the claim, completing the analysis of the algorithm.

Proof of Claim 1: The probability of reaching n from $n-j$ within $3j \leq 3n$ steps is at least $\binom{3j}{j}\left(\frac{2}{3}\right)^{j}\left(\frac{1}{3}\right)^{2j}$, as this last expression corresponds to the probability of making exactly $2j$ steps to the right and j steps to the left. Substituting Stirling's approximation for the binomial coefficient (next section), we get that the probability is at least:

$$\frac{c}{\sqrt{j}}\frac{27^j}{4^j}\frac{2^j}{3^j}\frac{1}{9^j} \geq \frac{c}{\sqrt{n}}\frac{1}{2^j}.$$

5 Stirling's approximation

Stirling's approximation for the factorial is:

$$\sqrt{2\pi n}\left(\frac{n}{e}\right)^n \leq n! \leq 2\sqrt{2\pi n}\left(\frac{n}{e}\right)^n.$$

In particular, if $m \geq \alpha n$ for $\alpha \geq 1$, then for $n \geq 1$:

$$\binom{m}{n} \geq \frac{c}{\sqrt{n}}\left(\frac{\alpha^\alpha}{(\alpha-1)^{\alpha-1}}\right)^n,$$

where the constant c depends on α.

While the proof of Stirling's approximation needs some work, it is easier to see why $n!$ is asymptotically close to $\left(\frac{n}{e}\right)^n$. Write $\log n! \sim \sum_{i=1}^{n} \log i \sim \int_1^n \log x \sim n\log n - n$.