Lecture 10. The Law of Iterated Logarithm. Randomized primality testing. Markov chains and 2-SAT

Martin Klazar

December 8, 2020

The Law of Iterated Logarithm

is a fundamental result in classical probability theory concerning infinitely many independent coin tosses, and a follow-up of the results concluding the last lecture. These are infinitely many events A_n, $n \in \mathbb{N}$, in a probability space $P = (\Omega, \Sigma, \Pr)$ such that for every finite set $I \subset \mathbb{N}$,

$$\Pr\big(\bigcap_{n\in I} A_n\big) = \big(\tfrac{1}{2}\big)^{|I|} .$$

We learned in Lecture 8 that it takes an uncountable probability space P (with uncountable set Ω) to accommodate infinitely many such events. We associate with each A_n its *modified indicator RV*

$$X_n(\omega) = \begin{cases} 1 & \dots \quad \omega \in A_n \\ -1 & \dots \quad \omega \in \Omega \setminus A_n . \end{cases}$$

Theorem (LIL). *Suppose that A_n, $n \in \mathbb{N}$, are independent coin tosses, X_n are their respective modified indicator RVs and $S_n := X_1 + X_2 + \cdots + X_n$. Then for any $\varepsilon > 0$,*

$$\Pr\Big(\tfrac{S_n}{\sqrt{(n/2)\log(\log n)}} > 1+\varepsilon \text{ for infinitely many } n\Big) = 0$$

and

$$\Pr\Big(\tfrac{S_n}{\sqrt{(n/2)\log(\log n)}} > 1-\varepsilon \text{ for infinitely many } n\Big) = 1 .$$

Thus if we perform infinitely many independent tosses of a fair coin (we can do it in our minds but not in reality), count $+1$ point for each head and -1 point for each tail, then we encounter almost surely

the event that infinitely often after n tosses the score is larger than $0.99\sqrt{(n/2)\log(\log n)}$, but we encounter almost never the event that infinitely often after n tosses the score is $> 1.01\sqrt{(n/2)\log(\log n)}$. Deviations from the expected zero score are unavoidable to certain degree, but no larger deviation, larger by any little bit, occurs.

The LIL is due to A. Ja. Chinčin in 1924 and A. N. Kolmogorov in 1929.

• *Aleksandr Ja. Chinčin (1894–1959)* was a Soviet mathematician working in probability theory and in number theory. He wrote the beautiful book *Three Pearls of Number Theory* (first published in Russian in Moscow and Leningrad in 1947) on van der Waerden's theorem, Mann's theorem and an elementary solution of Waring's problem. In the age of Wikipedia it suffices to mention just these three keywords.

• *Andrej N. Kolmogorov (1903–1987)* was, to quote Wikipedia, a Soviet mathematician who made significant contributions to the mathematics of probability theory, topology, intuitionistic logic, turbulence, classical mechanics, algorithmic information theory and computational complexity. He is the author of the currently accepted and widely used formalization of probability theory, published in 1933 in his treatise *Grundbegriffe der Wahrscheinlichkeitsrechnung* (Fundamental notions of the probability calculus).

Randomized primality testing

In the longish statement of the next theorem — we will not prove it — we describe a randomized primality test, due to M. O. Rabin and D. E. Knuth, and its properties.

Theorem (probable primes). *There is an algorithm*

$$\mathcal{A}(n,\,x)\colon \{(n,\,x)\in\mathbb{N}^2 \mid n>1 \text{ and is odd, } 1<x<n\}\to\{0,\,1\}\,,$$

described below, with the variables $k, n, q, y \in \mathbb{N}$, $j \in \mathbb{N}_0$ and the failure parameter $x \in \mathbb{N}$, and with the output 0 interpreted as "n is definitely not prime" and 1 as "n is probably prime". The algorithm $\mathcal{A}$ has the following steps.

1. *Main input: an odd number $n = 1 + 2^k q > 1$, q odd, to be tested for primality.*

2. *Input the failure parameter* $x \in \mathbb{N}$, $1 < x < n$.

3. *Exponentiation:* $j := 0$ *and take the* $y \in [n]$ *such that* $y \equiv x^q$ *modulo* n.

4. *Done? [now* $y \equiv x^{2^j q}$ *modulo* n.*] If* $j = 0$ *and* $y = 1$, *or if* $y = n - 1$, *output "*n *is probably prime" and terminate.*

5. *Increase* j*:* $j := j + 1$. *If* $j < k$, *set* $y := y^2$ *modulo* n, $y \in [n]$, *and go to step* 4.

6. *Not prime: output "*n *is definitely not prime" and terminate.*

This algorithm terminates on every input (n, x) *in time polynomial in* $\log n$.
God's view: If n *is prime, then for every* x *the algorithm correctly outputs "*n *is probably prime". If* n *is composite, then for some less than* $\frac{n-2}{4}$ *numbers* x *with* $1 < x < n$, *known to me, the algorithm fails and outputs "*n *is probably prime", but for other values of* x *it correctly outputs "*n *is definitely not prime".*
User's view: The answer of the algorithm "n *is definitely not prime" is always correct, for any* x. *The other answer "*n *is probably prime" may be incorrect for some less than* $\frac{n-2}{4}$ *numbers* x *with* $1 < x < n$ *but unfortunately I do not know which are these* x.

Exercise. Prove that if n is a prime number then for any $x \in \mathbb{N}$ with $1 < x < n$ the algorithm $\mathcal{A}$ outputs correctly that "n is probably prime". □

As we said, the algorithm is due to Rabin and Knuth, in cca 1977, and the above steps 1–6 are (with a small modification) as on p. 395 in *TAOCP, Volume 2*. I have to mention here that a deterministic polynomial-time test of primality, more complicated than the above algorithm (and therefore not really used in practice), was found in 2004 by M. Agrawal, N. Kayal and N. Saxena.

• *Michael O. Rabin* is an Israeli computer scientist. He was born in Breslau in Germany, which is today Wrocław in Poland.

• *Donald E. Knuth* is an American computer scientist. He is the author of the computer typesetting program `TeX` (in a variant of it,

LaTeX, all these lectures are written) and of the series of books *The Art of Computer Programming*, of which the volumes 1, 2, 3 and 4A have been already published.

The algorithm $\mathcal{A}$ shows its usefulness when it runs several times on the same main input n but different numbers x, which we formalize as follows. For every $r \in \mathbb{N}$ we have the algorithm $\mathcal{A}_r$ with input $(n, x_1, \ldots, x_r)$, where (n, x_i) are r inputs of $\mathcal{A}$, and the same output $\{0, 1\}$ as $\mathcal{A}$. The output is determined by

$$\mathcal{A}_r(n,\, x_1,\, \ldots,\, x_r) := \min(\mathcal{A}(n,\, x_1),\, \mathcal{A}(n,\, x_2),\, \ldots,\, \mathcal{A}(n,\, x_r))$$

— we run $\mathcal{A}$ r times on the inputs $(n, x_1), \ldots, (n, x_r)$ and output 0 ("n is definitely not prime") iff in at least one of the runs we get $\mathcal{A}$-output 0, and else we output 1 ("n is probably prime"). Let $I_n := \{2, 3, \ldots, n-1\}$. By the properties of $\mathcal{A}$ given in the above theorem we see that $\mathcal{A}_r$ runs in time polynomial in $\log n$ (with the implicit constant depending of course on r) and has the fraction of inputs on which it fails

$$\frac{|\{(x_1,\, \ldots,\, x_r) \in I_n^r \mid \mathcal{A}_r \text{ fails on } (n,\, x_1,\, \ldots,\, x_r)\}|}{|I_n^r| = (n-2)^r} < \left(\frac{1}{4}\right)^r$$

($\mathcal{A}_r$ fails on $(n, x_1, \ldots, x_r)$ iff $\mathcal{A}$ fails on each $(n, x_1), \ldots, (n, x_r)$). This deterministic inequality is usually interpreted probabilistically, one says something like: so by running $\mathcal{A}$ on n and sufficiently many random numbers x, $1 < x < n$, we can make the probability of failure of the algorithm as close to 0 as we wish. The problem with it is what do the "random numbers x" rigorously (not poetically) mean: we can input in $\mathcal{A}$ only numbers, not probability distributions. For example, why not let $\mathcal{A}$ run simply on n and $x = 2, 3, \ldots, r+1$? I leave this problem at that and move to the next topic.

Markov chains

• *Andrej A. Markov (1856–1922)*, after whom Markov chains are named, was a Russian mathematician. His son A. A. Markov (1903–1979) with identical name was a notable logician who introduced so called normal algorithms, a formalization of the notion of an algorithm.

Suppose that $p_{i,j} \geq 0$, $i, j \in \mathbb{N}_0$, are real numbers such that $\sum_j p_{i,j} = 1$ for every $i \in \mathbb{N}_0$. We say that a sequence of discrete RVs $X_0, X_1, \ldots$ such that $X_t \in \mathbb{N}_0$ forms a *Markov chain M (with transition probabilities $p_{i,j}$)* if for every $t \in \mathbb{N}$ and every $a_k \in \mathbb{N}_0$,

$$\begin{aligned}&\Pr(X_t = a_t \,|\, X_{t-1} = a_{t-1} \wedge X_{t-2} = a_{t-2} \wedge \cdots \wedge X_0 = a_0)\\ &= \Pr(X_t = a_t \,|\, X_{t-1} = a_{t-1})\\ &= p_{a_{t-1}, a_t}\end{aligned}$$

(the $\wedge$ are really $\cap$) whenever the initial conditional probability is defined, i.e. whenever

$$\Pr(X_{t-1} = a_{t-1} \wedge X_{t-2} = a_{t-2} \wedge \cdots \wedge X_0 = a_0) > 0 \,.$$

Thus M is a process with very short memory: it only remembers what happened in the previous step but not what happened before. Since I was giving this basic definition incorrectly in my oral lectures[1], I repeat it again next time. Then I also discuss the question of existence of Markov chains.

So for $i, j \in \mathbb{N}_0$ we call the probabilities $p_{i,j} = \Pr(X_t = j \,|\, X_{t-1} = i)$ (the right side may not be defined) the *transition probabilities*. The values of X_t in $\mathbb{N}_0$, or in $[n]_0 = \{0, 1, \ldots, n\}$ for *finite Markov chains*, are the *states*. We collect the transition probabilities in the *transition matrix*

$$P = (p_{i,\,j}) \in [0,\,1]^{\mathbb{N}_0 \times \mathbb{N}_0} \quad \text{or} \quad P = (p_{i,\,j}) \in [0,\,1]^{[n]_0 \times [n]_0} \,.$$

Exercise. Why do these matrices have every row sum 1? Because they were so defined. Why were they so defined? □

The *distribution* of M at time $t \in \mathbb{N}_0$ is described by the $1 \times \mathbb{N}_0$ or $1 \times (n+1)$ row vector

$$\overline{p}(t) = \big(p_i(t) \mid i \in \mathbb{N}_0 \text{ or } i \in [n]_0\big)$$

where $p_i(t) := \Pr(X_t = i)$. The entries in $\overline{p}(t)$ sum up to 1 (these are probabilities of events partitioning Ω).

[1] I was parroting, after some textbooks, that "A sequence of random variables $X_0, X_1, X_2, \ldots$ is a Markov chain if $\ldots$". This is the *wrong* way to begin the definition. Transition probabilities $p_{i,j}$ have to be given first, in general they cannot be determined from the X_t because the corresponding conditional probabilities may not be defined. Imprecisely formulated definitions of Markov chains in this style appear not so rarely in the literature.

We claim that for every $t \in \mathbb{N}$ the vector $\overline{p}(t)$ is related to the vector $\overline{p}(t-1)$ via the multiplication of (possibly infinite) matrices

$$\overline{p}(t) = \overline{p}(t-1)P = \overline{p}(t-1) \cdot P \ .$$

Indeed, for any $j \in \mathbb{N}_0$ we compute that

$$\begin{aligned}
(\overline{p}(t))_{1,\,j} &= p_j(t) = \Pr(X_t = j) \\
&= \sum_i \Pr(X_t = j \wedge X_{t-1} = i) \\
&= \sum_{i,\,\Pr(X_{t-1}=i)>0} \Pr(X_t = j \wedge X_{t-1} = i) \\
&= \sum_{i,\,\Pr(X_{t-1}=i)>0} \Pr(X_t = j \,|\, X_{t-1} = i) \cdot \Pr(X_{t-1} = i) \\
&= \sum_{i,\,\Pr(X_{t-1}=i)>0} p_{i,\,j} \cdot p_i(t-1) \\
&= \sum_i p_i(t-1) \cdot p_{i,\,j} = (\overline{p}(t-1) \cdot P)_{1,\,j} \ .
\end{aligned}$$

Watch carefully how the problem that conditional probability may not be defined was dealt with. So after $m \in \mathbb{N}_0$ steps (units of time) the distribution $\overline{p}(t)$ of M evolves in the distribution

$$\overline{p}(t+m) = \overline{p}(t) \cdot P^m \ .$$

Suppose that $X_0, X_1, \ldots$ is a Markov chain with transition matrix $P = (p_{i,j})$ and that $t \in \mathbb{N}_0$ and $m \in \mathbb{N}$. We claim that

$$(P^m)_{i,\,j} = \Pr(X_{t+m} = j \,|\, X_t = i)$$

whenever the right side is defined. This can be proved by induction on m: for $m = 1$ we are done by the definition of P, and the induction step $m \rightsquigarrow m+1$ follows by a computation similar to the one displayed above.

Exercise. Also, for every $t \in \mathbb{N}_0$ and any states $a_0, a_1, \ldots, a_t$,

$$\begin{aligned}
&\Pr(X_0 = a_0) p_{a_0,a_1} p_{a_1,a_2} \ldots p_{a_{t-1},a_t} \\
&= \Pr(X_t = a_t \wedge X_{t-1} = a_{t-1} \wedge \cdots \wedge X_0 = a_0) \ .
\end{aligned}$$

□

Clearly it will be useful to have a combinatorial interpretation of entries in powers of matrices. Let $m \in \mathbb{N}_0$ and $A \in \mathbb{R}^{n\times n}$, $n \in \mathbb{N}$, be any matrix. We assign to A the weighted directed graph $D_A = ([n], E, h)$, $E \subset [n]^2$ and $h\colon E \to \mathbb{R}$, defined by

$$e = (i,\, j) \in E \iff A_{i,\,j} \neq 0, \quad \text{and} \quad h(e) = h(i,\, j) := A_{i,\,j}\ .$$

For $u, v \in [n]$ a *u-v walk w in D_A with length $m \in \mathbb{N}_0$* is any sequence of vertices $w = (v_0, v_1, \ldots, v_m) \subset [n]$ such that $v_0 = u$, $v_m = v$ and $(v_{i-1}, v_i) \in E$ for every $i = 1, 2, \ldots, m$. We define its weight $h(w)$ by

$$h(w) := \prod_{i=1}^{m} h(v_{i-1},\, v_i) = \prod_{i=1}^{m} A_{v_{i-1},\, v_i}\ .$$

If $m = 0$ we set $h(w) = 1$. The proof of the next proposition is easy and therefore is left as an exercise.

Proposition (walks and powers). *For every $m \in \mathbb{N}_0$, $n \in \mathbb{N}$, every matrix $A \in \mathbb{R}^{n\times n}$, and any indices/vertices $i, j \in [n]$,*

$$(A^m)_{i,\,j} = \sum_{\substack{w \text{ is} \\ \text{an } i\text{-}j \text{ walk} \\ \text{in } D_A \text{ with} \\ \text{length } m}} h(w)$$

where the empty sum, which appears for instance if $m = 0$ and $i \neq j$, is defined as 0.

2-SAT problem and Markov chains

(This final part of L10 will not be examined, as of January 13 it is in an unconvincing state; I rework it later.)

Suppose that $x_1, \ldots, x_n$ are Boolean variables, ranging in $\{0, 1\}$, and $F = F(x_1, \ldots, x_n)$ is a Boolean 2-SAT formula of the form

$$F(x_1,\, \ldots,\, x_n) = \bigwedge_{j=1}^{k} (y_j \vee z_j)$$

where each $y_j, z_j \in \{x_1, \ldots, x_n, \overline{x_1}, \ldots, \overline{x_n}\}$. Here $\overline{x_i}$ negates the variable x_i, each disjunction $y_j \vee z_j$ is a *clause of F*, the y_j, z_j for $j =$

$1, 2, \ldots, k$ are *literals*, and the "2" means that each clause contains exactly two literals. The *2-SAT problem* asks to decide efficiently if there exists a *satisfying assignment* $x_i \in \{0,1\}$ such that $F(x_1, \ldots, x_n) = 1$ for it, and if it exists then to find it efficiently. For example, for $n = 4$ and

$$F = (x_4 \vee \overline{x_1}) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (x_4 \vee \overline{x_3})$$

we have $F(1,1,1,1) = 0$, but a satisfying assignment for this formula exists because, for instance, $F(1,0,0,1) = 1$.

The **randomized 2-SAT algorithm** for the 2-SAT problem in Mitzenmacher and Upfal, p. 171, has the following four steps.

1. Start with an arbitrary assignment $S_0 \colon \{x_1, \ldots, x_n\} \to \{0,1\}$.
2. Repeat up to $2mn^2$ times, $m \in \mathbb{N}$, terminating if all clauses are satisfied:
   (a) Choose arbitrary clause that is not satisfied.
   (b) Choose uniformly at random one of the literals in the clause and switch the value of its variable.
3. If a satisfying assignment has been found, return it.
4. Else, return that F is unsatisfiable.

We analyze it, after Mitzenmacher and Upfal (pp. 171–174).

Let $S \colon \{x_1, \ldots, x_n\} \to \{0,1\}$ be a fixed satisfying assignment for F and X_i be the RV recording the number of variables in the current assignment that have after step i the same value as in S. Clearly $\Pr(X_{i+1} = 1 \,|\, X_i = 0) = 1$ and for $j \in [n-1]$,

$$\Pr(X_{i+1} = j+1 \,|\, X_i = j) \geq \frac{1}{2} \text{ and } \Pr(X_{i+1} = j-1 \,|\, X_i = j) \leq \frac{1}{2}\,.$$

$X_0, X_1, \ldots$ is not necessarily a Markov chain. We define a Markov chain $Y_0, Y_1, \ldots$ with states $[n]_0$ by $Y_0 := X_0$, $\Pr(Y_{i+1} = 1 \,|\, Y_i = 0) = 1$, and for $j \in [n-1]$,

$$\Pr(Y_{i+1} = j+1 \,|\, Y_i = j) = \frac{1}{2} \text{ and } \Pr(Y_{i+1} = j-1 \,|\, Y_i = j) = \frac{1}{2}\,.$$

It follows that

$$\mathbb{E}\big(\substack{\text{time to reach } n \text{ in the process } X_t, \\ \text{starting from the distribution } X_0}\big) \le \mathbb{E}(\dots\ Y_t\ \dots) = \sum_{t=0}^{\infty} t \cdot \Pr(Y_t = n)\ .$$

We can think of the Markov chain Y_t also as a random walk on the undirected path with vertices $[n]_0$ and edges $\{i, i-1\}$, $i \in [n]$; from each vertex we go with the same probability to its neighbor ($i = 0$ and $i = n$ have just one neighbor, any other vertex i has two). Let $Z_j \in \mathbb{N}_0$, for $j \in [n-1]$, be the RV recording the number of steps it takes to reach by the walk the endvertex n from the vertex j. From the vertex j we go with equal probability $\frac{1}{2}$ either to $j-1$ or to j. Hence

$$\mathbb{E}Z_j = \mathbb{E}\big(\tfrac{1}{2}(1 + Z_{j-1}) + \tfrac{1}{2}(1 + Z_{j+1})\big)\ .$$

If $h_j := \mathbb{E}Z_j$, we have by linearity of expectation that

$$h_j = \tfrac{1}{2}(h_{j-1} + h_{j+1}) + 1\ .$$

We have got the system of linear equations ($j \in [n-1]$)

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

Lemma. *The above equation imply that for any $j \in [n-1]_0$ one has the relation $h_j = h_{j+1} + 2j + 1$. Hence*

$$h_0 = h_1 + 1 = h_2 + 1 + 3 = \cdots = 0 + \sum_{i=0}^{n-1}(2i+1) = n^2\ .$$

Proof. By induction on j. For $j = 0$ the relation holds by the first equation in the system. For $j \ge 1$ we have by the second equation in the system and by induction that

$$h_{j+1} = 2h_j - h_{j-1} - 2 = 2h_j - (h_j + 2j - 1) - 2 = h_j - 2j - 1\ ,$$

which is the relations for j. □

Thus we have the following

Corollary. *If a 2-SAT formula with n variables has a satisfying assignment and if the above 2-SAT algorithm runs until it finds a satisfying assignment, then*

$$\mathbb{E}\left(\begin{smallmatrix}\text{the number of steps until the}\\ \text{algorithm finds a satisfying assignment}\end{smallmatrix}\right) \leq n^2 \ .$$

Theorem. *The 2-SAT algorithm always returns a correct answer if F is unsatisfiable. If F is satisfiable, then with probability at least $1 - 2^{-m}$ algorithm returns a satisfying assignment. Else it incorrect returns that F is unsatisfiable*

Proof. The case when there is no satisfying assignment is clear. Let F be satisfiable. We divide executions of the algorithm in m segments with length $2n^2$. The Corollary implies that

$$\mathbb{E}\left(\begin{smallmatrix}\text{no satisfying assignment}\\ \text{found in the } i\text{-th segment}\end{smallmatrix} \,\middle|\, \begin{smallmatrix}\text{no satisfying assignment found}\\ \text{in the segments } 1, 2, \ldots, i-1\end{smallmatrix}\right) < \frac{1}{2}$$

because if Z is the RV recording the number of steps from the start of the i-th segment until the algorithm finds a satisfying assignment, then $\mathbb{E}Z \leq n^2$ by the Corollary, and therefore by Markov's inequality

$$\Pr(Z > 2n^2) \leq \frac{\mathbb{E}Z}{2n^2} = \frac{n^2}{2n^2} = \frac{1}{2} \ .$$

Hence

$$\Pr\left(\begin{smallmatrix}\text{the algorithm fails to find a satisfying}\\ \text{assignment after } m \text{ segments}\end{smallmatrix}\right) \leq (1/2)^m$$

and the theorem is proven. □

We conclude with the remark that the 2-SAT problem is solvable in polynomial time, but the analogous 3-SAT problem is NP-complete.

Thank you!

(semifinal version — the last part is to be rewritten — of January 13, 2021)