Institut for Matematik og Datalogi
Syddansk Universitet

September 12, 2023
JBJ

DM551/MM851 – Fall 2023 – Weekly Note 3

Stuff covered in Week 37:

- Rest of Rosen 6.5.
- Rosen 7.1
- Rosen 7.2.1-7.2.7 and 7.2.9.

Lectures in Week 38

- Section 7.3 pages 494-497 (This is the only part of Section 7.3 that is pensum.
- Rosen 7.4
- Notes by me on Random variables and the probabilistic method. See below

Exercises in Week 38

Remember that it will not necessarily be all exercises that are solved in the class so if there is one or more that you want to see, you should inform your TA when you meet in class.

- Section 6.5: 38, 40, 43, 48, 58
- Section 7.1: 6, 12, 17, 36, 45
- Section 7.2: 15.
  This simple but extremely important in-equality is called the **union-bound** and is used very often.
- Section 7.2: 12, 23, 30,36,38
- Any remaining exercises from Week 37 that you would like to discuss.

Notes on indicator random variables

Let S be a sample space and let A be an event in S ($A \subseteq S$). Let X_A be the random variable which takes the value 1 when A occurs and 0 when A does not occour. That is,

$$X_A(s) = \begin{cases} 1 & \text{If } s \in A \\ 0 & \text{If } s \notin A \end{cases} \tag{1}$$

We say that X_A is the **indicator variable** for the event A. When A is clear from the context we drop the subscript A.

Theorem 1 *Let A be an event in a sample space S and let X be the indicator variable for A. Suppose the probability that A occurs is p ($p(A) = p$). Then expected value and variance of X is given by $E(X) = p$ and $V(X) = p(1-p)$.*

Proof:

$$\begin{aligned} E(X) &= 0 \cdot p(X=0) + 1 \cdot p(X=1) \\ &= 0 + 1 \cdot p \\ &= p. \end{aligned}$$

For the variance we want to use the formula $V(X) = E(X^2) - (E(X))^2$. This is easy to apply here since every indicator variable X satisfies that $X(s) = X(s)^2$ for all $s \in S$. Now we get

$$\begin{aligned} V(X) &= E(X^2) + (E(X))^2 \\ &= E(X) - (E(X))^2 \\ &= E(X)(1 - E(X)) \\ &= p(1-p). \end{aligned}$$

$\diamond$

Why are indicator variables useful? Because they allow us to simplify several proofs and to analyze probability of certain events with much less effort. Here is an example and there will be many more in the following weeks.

Theorem 2 *Consider n independent Bernoulli trials each of which have probability of success equal to p. Let X be the random variable the denotes the number of successes in these n trials. Then $E(X) = np$ and $V(X) = np(1-p)$.*

Proof: Let X_i, $i = 1, 2, \ldots, n$ be the indicator random variable for the event that the ith trial is a success (so $p(X_i = 1) = p$ for all $i = 1, 2, \ldots, n$). Then $X = \sum_{i=1}^n X_i$ so by linearity of expected values we get $E(X) = \sum_{i=1}^n E(X_i) = \sum_{i=1}^n p = np$. Furthermore, as X_i and X_j are independent random variables (the experiments are independent) for all distinct i, j we have $V(X) = \sum_{i=1}^n V(X_i) = \sum_{i=1}^n p(1-p) = np(1-p)$. $\diamond$

3 Notes on the probabilistic method

The following important examples of the usefulness of the probabilistic method are not in Rosen (Markov's inequality is covered in Exercise 27 of Section 7.4):

Markov's Inequality For any non-negative random variable X on a sample space S,

$$p(X \geq t) \leq \frac{E(X)}{t}$$

Proof: By Theorem 1 page 464 in Rosen we have $E(X) = \sum_{i \in X(S)} p(X = i)i$. Since $X(s) \geq 0$ for every $s \in S$ this gives
$E(X) \geq \sum_{i \in X(S), i \geq t} p(X = i)i \geq t \sum_{i \in X(S), i \geq t} p(X = i) = p(X \geq t)t$. $\diamond$

First Moment Principle If $E(X) \leq t$ then $p(X \leq t) > 0$.

Proof: Suppose $p(X \leq t) = 0$. Then we have
$E(X) = \sum_{i \in X(S)} p(X = i)i \geq \sum_{i \in X(S), i > t} p(X = i)i > t \sum_{i \in X(S), i \geq t} p(X = i) = t$, where we used that $p(X \leq t) = 0$. $\diamond$

A **boolean variable** x is a variable that can assume only two values 0 and 1. The **sum** of boolean variables $x_1 + x_2 + \ldots + x_k$ is defined to be 1 if at least one of the x_i's is 1 and 0 otherwise. The **negation** $\overline{x}$ of a boolean variable x is the variable that assumes the value $1 - x$. Hence $\overline{\overline{x}} = x$. Let X be a set of boolean variables. For every $x \in X$ there are two **literals**, over x, namely, x itself and $\overline{x}$. A **clause** C over a set of boolean variables X is a sum of literals over the variables from X. The **size** of a clause is the number of literals it contains. For example, if u, v, w are boolean variables with values $u = 0, v = 0$ and $w = 1$, then $C = (u + \overline{v} + \overline{w})$ is a clause of size 3, its value is 1 and the literals in C are u, $\overline{v}$ and $\overline{w}$. An assignment of values to the set of variables X of a boolean expression is called a **truth assignment**. If the variables are $x_1, \ldots, x_k$, then we denote a truth assignment by $t = (t_1, \ldots, t_k)$. Here it is understood that x_i will be assigned the value t_i for $i \in \{1, 2, \ldots, k\}$.
The **satisfiability problem**, also called **SAT**, is the following problem. Let $X = \{x_1, \ldots, x_n\}$ be a set of boolean variables and let $C_1, \ldots, C_m$ be a collection of clauses for which every literal is over X. Decide if there exists a truth assignment $t = (t_1, \ldots, t_n)$ to the variables

in X such that the value of every clause will be 1. This is equivalent to asking whether or not the boolean expression $\mathcal{F} = C_1 * \ldots * C_m$ can take the value 1. Depending on whether this is possible or not, we say that $\mathcal{F}$ is **satisfiable** or **unsatisfiable**. Here '$*$' stands for **boolean multiplication**, that is, $1 * 1 = 1$, $1 * 0 = 0 * 1 = 0 * 0 = 0$. For a given truth assignment $t = (t_1, \ldots, t_n)$ and literal q we denote by $q(t)$ the value of q when we use the truth assignment t (i.e., if $q = \overline{x_3}$ and $t_3 = 1$, then $q(t) = 1 - 1 = 0$).

To illustrate the definitions, let $X = \{x_1, x_2, x_3\}$ and let $C_1 = (\overline{x_1} + \overline{x_3})$, $C_2 = (x_2 + \overline{x_3})$, $C_3 = (\overline{x_1} + x_3)$ and $C_4 = (x_2 + x_3)$. Then it is not difficult to check that $\mathcal{F} = C_1 * C_2 * C_3 * C_4$ is satisfiable and that taking $x_1 = 0, x_2 = 1, x_3 = 1$ we obtain $\mathcal{F} = 1$.

If all clauses have the same number k of literals, then we have an instance of k-**SAT** (the example above is an instance of 2-SAT). Generally k-SAT is a very difficult problem and you will see in the spring course "Complexity and Computability" that is one of the so-called $\mathcal{NP}$-complete problems for which no-one knows a polynomial algorithm.

Theorem A For every natural number k, every k-SAT formula with less than 2^k clauses is satisfiable.

Proof: Consider a random truth assignment which sets variable x_i to 1 with probability $\frac{1}{2}$ and to 0 with probability $\frac{1}{2}$ for $i = 1, 2, \ldots, n$. Note that by this assignment, each of the 2^n possible truth assignments are equally likely (they all have probability 2^{-n}).

Let X be the random variable defined on the set of all truth assignments which to a given truth assignment $t = (t_1, \ldots, t_n)$ assigns the value $X(t) =$ the number of clauses among $C_1, C_2, \ldots, C_m$ which are **not** satisfied by t. Similarly, for each clause C_i we let the random variable X_i take the value $X_i(t) = 1$ if t does **not** satisfy C_i and $X_i(t) = 0$ if t satisfies C_i. Thus $X(t) = X_1(t) + X_2(t) + \ldots + X_m(t)$. We call the X_i's **indicator random variables** and their expectations are easy to calculate:

$E(X_i) = p(X_i = 1)1 + p(X_i = 0)0 = p(X_i = 1) = 2^{-k}$, since the i'th clause evaluates to 0 precisely if all k literals are 0 and each of these are 0 with probability 1/2.

Now we get, by linearity of expectations

$E(X) = \sum_{i=1}^{m} E(X_i) = \sum_{i=1}^{m} 2^{-k} = 2^{-k} \sum_{i=1}^{m} 1 = m2^{-k} < 1$, since $m < 2^k$.

Hence, by Markov's inequality, $p(X \geq 1) \leq \frac{E(X)}{1} = E(X) < 1$ so $p(X = 0) > 0$. This shows that there is at least one of the 2^n truth assignments which satisfies all m clauses. $\diamond$

The bound on the number of clauses in Theorem A is best possible: suppose we have $n = k$ and all the 2^k clauses of size k over these variables (every clause contains each variable either with or without negation), then clearly this instance is not satisfiable, since no matter which truth assignment we take, some clause will have all literals evaluating to 0. But observe that removing just one we get a satisfiable instance by the theorem!

Using the same argument as above we get the following bound for general SAT (clauses may have any size):

Theorem B Let $\mathcal{F} = C_1 * C_2 * \ldots * C_m$ be an instance of SAT[1]. If we have $\sum_{i=1}^{m} 2^{-|C_i|} < 1$, then $\mathcal{F}$ is satisfiable. $\diamond$

Corollary For all $\epsilon > 0$ there exists a polynomial algorithm for solving any instance of SAT over n variables $x_1, x_2, \ldots, x_n$ in which all clauses have size at least ϵn.

Proof: Let $\epsilon > 0$ be given and let $\mathcal{F} = C_1 * C_2 * \ldots * C_m$ over the variables $x_1, x_2, \ldots, x_n$ satisfy that $|C_i| \geq \epsilon n$ for each $i \in \{1, 2, \ldots, m\}$. Suppose first that $m < 2^{\epsilon n}$. Then we have

$$\sum_{i=1}^{m} 2^{-|C_i|} \leq \sum_{i=1}^{m} 2^{-\epsilon n} = m2^{-\epsilon n} < 1$$

Hence it follows from Theorem B that $\mathcal{F}$ is satisfiable and our algorithm stops with a "yes". Clearly this can be checked in time polynomial in $|\mathcal{F}|$ since we just need to check whether the number of clauses is less than $2^{\epsilon n}$. **Note that in this case we do not find a satisfying truth assignment!** We just answer correctly that there **exists** one.

Now suppose that we found that there was at least $2^{\epsilon n}$ clauses. Then we simply check all the 2^n possible truth assignments to see whether one of these satisfies $\mathcal{F}$. If we find one that does, we stop and answer "yes" otherwise, after checking that none of them satisfy $\mathcal{F}$, we answer "no". The time required to do this is proportional to $2^n|\mathcal{F}|$, where $|\mathcal{F}|$ is the size of the formula $\mathcal{F}$ and hence of the input. Clearly $|\mathcal{F}| \geq 2^{\epsilon n}$ as all clauses have size at least 1 (in fact $|\mathcal{F}| \geq \epsilon n 2^{\epsilon n}$). Form this we get that $2^n \leq |\mathcal{F}|^{\frac{1}{\epsilon}}$ so the running time of our algorithm is proportional to $2^n|\mathcal{F}| \leq |\mathcal{F}|^{1+\frac{1}{\epsilon}}$ which is a polynomial in $|\mathcal{F}|$ because ϵ is a constant (when we have chosen it). $\diamond$

[1] Over variables $x_1, x_2, \ldots, x_n$ but they play no role in the argument.