Probability and Computing

Lecturers
Samuel Baguley, Andreas Goebel, Panagiotis Aivasiliotis

Notes
Simon Cyrani

Version
git: a1865ec-*
compiled: Tuesday 16th April, 2024 00:28

Abstract

The following lecture notes are my personal (and therefore unofficial) write-up for 'Probability and Computing' aka 'ProbComp', which took place in summer semester 2024 at Hasso-Plattner-Institut. I do not guarantee correctness, completeness, or anything else. Importantly, note that I willfully changed some specific notations, reordered some material, and left out parts that I didn't found worth typing down.

If you miss something, feel free to contribute in the repository!

Contents

Summary of lectures

Part I

Lecture notes

Lecture 1
Mo 15 Apr 2024

1 Probabilistic method

This section will introduce how probability can be used to solve problems that at first glance do not have anything to do with probability. Consider following combinatorial problem.

Definition 1.1 (Ramsay numbers)**.** We define R_k as the smallest integer n such that any graph with n vertices must contain either a clique or an independent set of size K. We call R_k the (symmetric) **Ramsey numbers**.

Let us first look at some examples to get a feel for what this section is about.

Example 1.2. The first few Ramsay numbers are given as

- $R_2 = 2$: Obviously, $R_k \geq k$. Furthermore, there are only two different graphs with two nodes, i.e. with or without an edge. In the former case, both nodes form a clique, in the latter they form an independet set of size 2.
- $R_3 = 6$: First, let us show $R_3 \geq 6$. Consider a cycle of 5 nodes. Then, there is no clique of size 3 (since there is no 3-cycle). Also, among any three nodes two nodes are connected by an edge, so there is no independent set. Therefore, this is a counterexample. Now, consider a graph with 6 nodes. Suppose there is no clique of size 3. Then it suffices to show that there is an independent set of 3 nodes. Indeed, with an ugly case distinction this is possible: If there are no cycles, the graph is bipartite, so there is an independent set of at least 3. Otherwise, there exists at least a 4-cycle, but no 3-cycle (i.e. chord-free). We can then select at least two independent nodes from the cycle, and if needed the missing third node from the non-cycle nodes such that they form an independent set.
- $R_4 = 18$. Trust me, we do not want the proof here.
- $R_5 \in [43, 48]$. The exact value is indeed still unknown!

As we can see, even for small k, it is not trivial to determine their Ramsay number. Instead, let us try to at least find some bound for their value.

Theorem 1.3. For every $k \geq 1$ holds $R_k > 2^{k/2}$.

Proof. Consider a uniform distribution over all graphs with n vertices (i.e. the Erdős–Rényi random graph $\mathcal{G}(n, \frac{1}{2})$). Each edge in particular exists with probability $\frac{1}{2}$. Now, have

a look at $p \coloneqq P(G \sim \mathcal{G}(n, \frac{1}{2})$ has a k-clique or k independent set). If we can show that this probability p is less than 1, then this means there is a graph with n vertices such that the property is *not* satisfied, and therefore $R_k > n$.

Let S be a k-tuple of vertices. S is per definition a k-clique if *all* or *none* of its edges is existent. Therefore, its probability of being either one is given as

$$P(S \text{ is } k\text{-clique or } k\text{-independent set}) = 2 \cdot \frac{1}{2^{\binom{k}{2}}}. \tag{1}$$

The total number of k-subsets given n vertices is given as $\binom{n}{k}$, so by basic properties of probability and binomial coefficients we see

$$p \leq \binom{n}{k} \cdot \frac{1}{2^{\binom{k-1}{2}}} \leq \frac{n^k}{k!} 2^{1-\frac{k^2-k}{2}} \tag{2}$$

For $n = 2^{k/2}$ the right-hand side reduces to $\frac{2^{k+2}}{k!}$, which can be shown easily to be smaller than 1 for $k \geq 3$. □

Notice how we suddenly imposed a probabilistic view on this presumably deterministic problem! This technique of using a suitable random model to demonstrate the existence and/or non-existence of certain properties is known as **Probabilistic Method**.

Before we have a look at another example, we need following lemma.

Lemma 1.4. Let X be a discrete random variable over a set Ω with $\mathbb{E}[X] = \mu$. Then, $P(X \geq \mu) > 0$ and $P(X \leq \mu) > 0$.

Proof. Assume $P(X \geq \mu) = 0$. Then $P(X = x) = 0$ for $x \geq \mu$. By definition and assumption therefore

$$\mathbb{E}[X] = \sum_{x \in X(\Omega)} xP(X = x) = \sum_{x < \mu} xP(X = x) \tag{3}$$

$$< \sum_{x < X(\Omega)} \mu P(X = x) = \mu \sum_{x \in X(\Omega)} P(X = x) = \mu. \tag{4}$$

This is a contradiction! □

Consider following problem.

Definition 1.5 (Max-Cut)**.** Given a graph $G = (V, E)$, find a set A maximizing the number of edges between A and $V \setminus A$. We call this the **Maximum Cut Problem**.

As it is usual in lectures of this kind, its canonical decision variant is indeed a NP-complete problem. Again, let us try instead to find a "good" cut.

Theorem 1.6 (Minimal Max-Cut)**.** Given a graph $G = (V, E)$ with $|E| = m$. There exists $A \subseteq V$ with at least $\frac{m}{2}$ cut size.

Proof. Choose A uniformly over $\mathcal{P}(V)$, i.e. every node is chosen with probability $\frac{1}{2}$. Then, every edge is with probability $\frac{1}{2}$ included in the cut, which happens iff exactly one of the nodes of the edge is in A. Let X be the number of cut edges, and X_e be the indicator variable for e being in the cut. By linearity of expectancy,

$$\mathbb{E}[X] = \mathbb{E}\left[\sum_{e \in E} X_e\right] = \sum_{e \in E} \mathbb{E}[X_e] = m \cdot \frac{1}{2}. \tag{5}$$

Using Lemma 1.4 there is positive probability for a choice of A having cut size at least $m/2$. □

You might have noticed that these are non-constructive results, so naturally following question emerges: Can we use these results to construct a solution? Indeed, there is a simple answer!

Definition 1.7 (Las-Vegas Algorithm)**.** Let T be the run-time of an algorithm and n its input size. We call a **Las-Vegas Algorithm** an algorithm which

1. always returns the correct answer with $\mathbb{E}[T] \in \mathcal{O}(n^k)$ for some k, or
2. has runtime always in $\mathcal{O}(n^k)$ for some k, and returns a correct answer with probability at least $\delta > 0$ (and otherwise returns no answer).

Remark 1.8. Both definitions are equivalent, which stems from the fact the second variant can be seen as a geometric process: Consider the number T of attempts until the algorithm returns a correct result. Then, $\mathbb{E}[T] = \frac{1}{1-\delta}$, so the total runtime is still $\mathcal{O}(n^k)$ in expectancy.

Turning back to the Max-Cut problem, let us transform our result of Theorem 1.6 into a Las-Vegas algorithm. Simply generate A randomly as constructed in the proof, and check if it has enough cut edges in polynomial time. The probability that this works is $P(X_A \geq \frac{m}{2}) = p$. However, this is just an abstract value - let us try to find a more meaningful way of expressing p:

$$\frac{m}{2} = \mathbb{E}[X_A] = \sum_{i < \frac{m}{2}} i \cdot P(X_A = i) + \sum_{i \geq \frac{m}{2}} i \cdot P(X_A = i) \tag{6}$$

$$\leq \left(\frac{m}{2} - 1\right) \sum_{i < \frac{m}{2}} P(X_A = i) + m \sum_{i \geq \frac{m}{2}} P(X_A = i) = \left(\frac{m}{2} - 1\right)(1 - p) + mp \tag{7}$$

Here we just upper-bound the corresponding first factors in each sum, and then use $P(X_A \geq \frac{m}{2}) = p$. Using some easy algebra, we deduce $p \geq \frac{1}{\frac{m}{2}+1}$, so our Las-Vegas approach seems to get gradually worse the more edges our graph has.

Interestingly, we do not even need a randomized algorithm to find a big cut. Instead, using probabilistic arguments we can construct a deterministic algorithm still running in polynomial time. This technique is known as **Derandomization**.

Algorithm 1: Find Big-Cut of $G = (V, E)$

$A \leftarrow \emptyset, B \leftarrow \emptyset$
for $k = 1, \dots, n$ **do**
  if $|\{(v_k, u) \in E \mid u \in A\}| \leq |\{(v_k, u) \in E \mid u \in B\}|$ **then**
    $A \leftarrow A + v_k$
  end
  else
    $B \leftarrow B + v_k$
  end
end
return A

The idea is to greedily decide for each vertex if we want it in our cut set or not based on a case distinction using conditional expectation.

Theorem 1.9. Given a graph $G = (V, E)$. Let $A \sim \mathcal{U}_{\mathcal{P}(V)}$, C_A the amount of cuts, and $x_1, \dots, x_n$ indicate if the vertices $v_1, \dots, v_n \in A$. Then Algorithm 1 satisfies following statements:

1. $\mathbb{E}[C_A \mid x_1, \dots, x_k] \leq \mathbb{E}[C_A \mid x_1, \dots, x_k, x_{k+1}]$ after iteration $k+1$ of the for-loop (such that x_i is fixed by the algorithm).
2. The algorithm runs in $\mathcal{O}(n+m)$ and returns a big cut of size at least $m/2$.

Proof. Notice for $1 \leq k < n$ by definition of conditional expectancy

$$\mathbb{E}[C_A \mid x_1, \dots, x_k] = \frac{1}{2}\mathbb{E}[C_A \mid x_1, \dots, x_k, x_{k+1} = 1] + \frac{1}{2}\mathbb{E}[C_A \mid x_1, \dots, x_k, x_{k+1} = 0].$$

So, at least one of the choices for x_{k+1} satisfy the required lower bound for the conditional expectancy. It remains to prove that our algorithm also chooses this value, i.e. the choice with larger condtional expectancy. Let us observe how the expected number of cut edges can change if we fix the position of vertex v_{k+1}. Consider following cases for an edge $e \in E$:

- e does not contain v_{k+1}. Then, by indepence, the expected value of its Is-Cut-Edge indicator X_e conditioned on all fixed vertices does not change by fixing v_{k+1} (i.e 1 or 0 if both vertices are determined, else $\frac{1}{2}$).
- e contains v_{k+1}, but the other vertex is not fixed. Again, aforementioned expected value does not change since there is still a $\frac{1}{2}$ probability for the other vertex being in A.
- e contains v_{k+1}, but the other vertex *is* fixed. Then we suddenly fix the value of X_e depending on the choice of x_{k+1}. In particular, this changes the conditional expectancy of X_e, increasing it to 1 or decreasing it to 0, and therefore the conditional expectancy on the number of cut edges changes in the same way.

In summary, the change in conditional expectancy by fixing x_{k+1} only depends on the neighborhood of fixed vertices of v_{k+1}. If there are more neighbors that are fixed to be A than not in A, then

$$\mathbb{E}\left[C_A \mid x_1, \ldots, x_k, x_{k+1} = 0\right] \geq \mathbb{E}\left[C_A \mid x_1, \ldots, x_k, x_{k+1} = 1\right],$$

otherwise the other way around, adhering to our algorithm design.

In fact, by induction it immediately follows that $\frac{m}{2} \leq \mathbb{E}\left[X_A\right] \leq \mathbb{E}\left[X_A \mid x_1, \ldots, x_n\right]$ for $x_1, \ldots, x_n$ being chosen according to the algorithm. So, our algorithm works, and has mentioned runtime. □

Part II
Appendix

A Exercise sheets

1. exercise sheet

Exercise 1.1. Placeholder.

Index