18.419 Random Walks and Polynomial-Time Algorithms 5 February 2002

Lecture 1: Introduction

Lecturer: Santosh Vempala *Scribe: Bobby Kleinberg*

In this course we will be considering the applications of random walks to the theory of algorithms. The following are some basic algorithmic areas where random walks are applied.

- optimization
- learning
- online scenarios, including data structures
- counting

In all of these areas, we will see that algorithmic problems may be reduced to the problem of sampling from particular probability distributions. The idea will be to use a random walk to efficiently sample from the desired distribution.

We will therefore be concerned with questions about the convergence rate of random walks to their stationary distribution. In general one bounds the convergence rate in terms of parameters called *isoperimetric coefficients*; these parameters also go by the name *conductance* or *expansion*. To see the relation between the convergence of random walks and isoperimetric inequalities, consider the case of a symmetric random walk on a graph G, i.e. the state of the random walk is a vertex v, and the state at the subsequent step is selected uniformly at random from the neighbors of v in G. If the graph contains a "bottleneck", i.e. a small cut which disconnects a large piece S from its complement $\overline{S}$ which is also large, then intuitively this should slow the convergence of the random walk, because states in S are unlikely to cross the small cut. This suggests that the isoperimetric coefficient should be defined as $\min_{S \subseteq V} \Pr(S \to \overline{S}|S)$, or more formally as

$$\min_{S \subseteq V} \frac{\sum_{i \in S, j \notin S} \pi_i \Pr(i \to j)}{\min(\sum_{i \in S} \pi_i, \sum_{i \notin S} \pi_i)}$$

where π_i denotes the probability of vertex i in the stationary distribution. This can be defined analogously in the continuous case.

1 Reduction of learning to sampling

We will now see our first example of reducing an algorithmic problem to a random sampling problem. In *learning theory*, one assumes there is an unknown function f belonging to a restricted class of functions. One is presented with samples $X^1, X^2, ...$ from the domain of the function, and one guesses at the values $f(X^i)$, learning after each guess whether it was right or wrong. The objective is to minimize the number of wrong guesses.

To take a specific example, suppose there's a fixed unknown vector $\vec{a} \in \mathbb{R}^n$, and our function f is defined by

$$f(\vec{x}) = \begin{cases} \text{True} & \text{if } \vec{a} \cdot \vec{x} \geq 0 \\ \text{False} & \text{if } \vec{a} \cdot \vec{x} < 0 \end{cases}$$

Assume the right answer has components $a_i \in \{-2^b, ..., 2^b\} \subset \mathbb{Z}$. Consider the following algorithm. At each iteration, choose a random $\vec{a}$ from those that have made no mistakes so far, and use that to make the next guess.

Theorem 1. E(number of mistakes) $\leq 2(b+1)n$.

Proof. If, on every step, you were to pick your answer according the majority vote of those $\vec{a}$ which have made no mistake so far, then every mistake would cut down the field of remaining voters by at least a factor of 2. As there are $2^{b(n+1)}$ voters at the outset, you would make at most $(b+1)n$ mistakes.

In the randomized algorithm above, every mistake has probability at least $1/2$ of halving the set of voters (in a sense which is made precise below), and the correct $\vec{a}$ is learned when the set of voters is halved $b(n+1)$ times. This is equivalent to tossing a weighted coin with bias $\geq \frac{1}{2}$ and waiting for $b(n+1)$ heads, so the expected number of tosses (i.e. mistakes) is less than or equal to $2b(n+1)$.

More precisely, let S_i denote the set of vectors which make no mistakes on the first i samples $X^1, \ldots, X^i$. Note that the probability of guessing correctly on step i is $\frac{|S_i|}{|S_{i-1}|}$. Consider the least k such that $|S_k| < \frac{1}{2}|S_0|$. The probability that the first mistake occurs on or after step k is

$$\frac{|S_1|}{|S_0|}\frac{|S_2|}{|S_1|} \cdots \frac{|S_{k-1}|}{|S_{k-2}|} = \frac{|S_{k-1}|}{|S_0|} \geq \frac{1}{2}$$

In other words, letting $i_1 < i_2 < \ldots < i_m$ denote the iterations on which mistakes are made, we have established that $|S_{i_1}| < \frac{1}{2}|S_0|$ with probability $\geq \frac{1}{2}$. Similarly, $|S_{i_{j+1}}| < \frac{1}{2}|S_{i_j}|$ with probability $\geq \frac{1}{2}$. This makes precise our earlier statement that "every mistake has probability at least $1/2$ of halving the set of voters."

Remark. This reduction of learning to sampling is general. The "vote with the majority" algorithm achieves the optimal worst-case number of mistakes, and the "sample randomly" algorithm achieves an expected number of mistakes which is a 2-approximation, via the same analysis as above.