College of Computer & Information Science
Northeastern University
Fall 2007
7 December 2007
CS G399: Algorithmic Power Tools I
Scribe: Xin Dong

Lecture Outline:

- Probability & Randomness

1 BPP

A probabilistic algorithm is an algorithm designed to use the outcome of a random process. Typically, such an algorithm would contain an instruction to "flip a coin" and the result of that coin flip would influence the algorithm's subsequent execution and output. Certain types of problems seem to be more easily solvable by probabilistic algorithms than by deterministic algorithms.

A probabilistic Turing machine M is a type of nondeterministic Turing machine where each nondeterministic step is called a coin-flip step and has two legal next moves. We assign a probability to each branch b of M's computation on input w as follows. Define the probability of branch b to be $Pr[b] = 2^{-k}$,

where k be the number of coin-flip steps that occur on branch b. Define the probability that M accepts w to be

$Pr[M \ \ accepts \ \ w] = \Sigma_{b \ \ is \ \ an \ \ accepting \ \ branch} Pr[b]$.

In addition, define

$Pr[M \ \ rejects \ \ w] = 1 - Pr[M \ \ accepts \ \ w]$.

For $0 \leq \epsilon \leq 1/2$ we say that M recognizes language A with error probability ϵ if

1. $w \in A$ implies $Pr[M \ \ accepts \ \ w] \geq 1 - \epsilon$, and

2. $w \notin A$ implies $Pr[M \ \ rejects \ \ w] \geq 1 - \epsilon$.

BPP is the class of languages that are recognized by probabilistic polynomial time Turing machines with an error probability of $1/3$.

2 PH

An alternating Turing machine is a nondeterministic Turing machine with an additional feature. Its states, except for q_{accept} and q_{reject}, divided into universal states and existential states. When we run an alternating Turing machine on an input string, we label each node of its nondeterministic computation tree with $\wedge$ or $\vee$, depending on whether the corresponding configuration contains a universal or existential state. We determine acceptance by designating a node to be accepting if it is labeled with $\wedge$ and all of its children are accepting or if it is labeled with $\vee$ and any of its children are accepting.

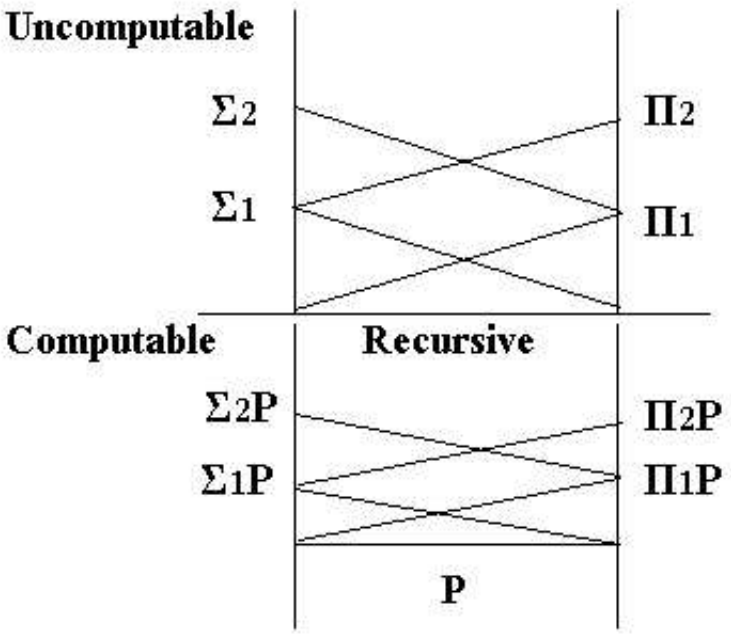

Figure 1: Kleene hierarchy

Let i be an integer greater than 0. A $\Sigma_i - alternating$ Turing machine is an alternating Turing machine that contains at most i runs of universal or existential steps, starting with existential steps. A $\Pi_i - alternating$ Turing machine is similar except that it starts with universal steps.

Let $\Sigma_i TIME(f(n))$ be the class of language that a $\Sigma_i - alternating$ Turing machine can recognize in $O(f(n))$ time. Similarly define the class $\Pi_i TIME(f(n))$ for $\Pi_i - alternating$ Turing machines. The polynomial time hierarchy is defined as the collection of classes

$\Sigma_i P = \cup_k \Sigma_i TIME(n^k)$ and

$\Pi_i P = \cup_k \Pi_i TIME(n^k)$.

The inclusive relationship among above classes is in Figure 1.

Define $PH = \cup_i \Sigma_i P = \cup_i \Pi_i P$.

3 RP, ZPP and $P^{\#P}$

RP is the class of languages that are recognized by probabilistic polynomial time Turing machines where inputs in the language are accepted with a probability of at least 1/2 and inputs not in the language are rejected with a probability of 1.

We have $P \subseteq ZPP = RP \cap coRP \subseteq BPP \subseteq PH$.

$\#P$ is the class of functions $f : \{0,1\}^* \to Z^{\geq 0}$ such that there exists a machine $M(\cdot,\cdot)$ running in polynomial time in the first input such that $\forall x, f(x) = \{y | M(x,y)\}$. $P^{\#P}$ is the class of languages decidable with the oracle access to $\#P$ functions.

By Toda's theorem, $PH \subseteq P^{\#P}$.

E is the class of exponential time but with linear exponent. We have: if any problem in E requires circuits of size $2^{\Omega(n)}$, then $BPP = P$. [IW97].

4 Probabilistic algorithms oi linear algebra

To determine whether $det(A) = 0$ such that A consists of variables is NP-hard. To determine whether $AB = C$, randomly pick $\vec{r}$ and check whether $AB\vec{r} = C\vec{r}$.

5 Deterministic hardness of volume estimation

For convex object, volume estimation equals to counting and is very hard. Under an oracle model, it is difficult to compute deterministically but can be done efficiently with randomness. Here, we show a lower bound for any deterministic algorithm.

The problem: Given n and convex body $K \in R^n$ such that $B(0,r) \subseteq K \subseteq B(0,R)$ for some $R > r > 0$. Compute the volume of K. Here, the convex body is determined by a membership oracle. Without loss of generality, we assume $R = 1$.

Claim: For any polynomial $m = f(n)$ time algorithm which gives the volume $vol(K)$ of the convex K and $p_1, \cdots, p_m \in K$, $vol(conv(< p_i >))/vol(B(1)) \leq m/(2-\epsilon)^n$.

Proof of claim: For each p_i, construct the ball B_i such that $\overrightarrow{op_i}$ is diameter. Let $K' = conv(< p_i >)$. Define $closure(K') = \{u | \exists v \in K', \overrightarrow{ov} \ \ subtends \ \ > \pi/2 \ \ at \ \ u\}$. Consider any $x \notin \cup_i B_i$. Because $\forall v \in K'$, we have $v = \Sigma_i \alpha_i p_i$ such that $\forall i, \alpha_i \geq 0$ and $\Sigma_i \alpha_i = 1$. Therefore $< v - x, -x >=< \Sigma_i \alpha_i p_i - x, -x >=< \Sigma_i \alpha_i p_i - \Sigma_i \alpha_i x, -x >= \Sigma_i \alpha_i < p_i - x, -x >> 0$. So $\overrightarrow{ov} \ \ subtends \ \ < \pi/2 \ \ at \ \ x$. That is if $x \notin \cup_i B_i$, then $x \notin closure(K')$. So $closure(K') \subseteq \cup_i B_i$.

Consider any $p \in K'$. We have $p = \Sigma_i a_i p_i$ such that $\forall i, a_i \geq 0$ and $\Sigma_i \alpha_i = 1$. Therefore $0 =< 0, -p >=< p - \Sigma_i a_i p, -p >=< \Sigma_i a_i p_i - \Sigma_i a_i p, -p >=< \Sigma_i a_i (p_i - p), -p >= \Sigma_i a_i < p_i - p, -p >$. Then there exists i such that $< p_i - p, -p >< 0$. That is $p \in closure(K')$. So $K' \subseteq closure(K')$.

In summary, we have $vol(conv(< p_i >)) = vol(K') \leq vol(closure(K')) \leq vol(\cup_i B_i) \leq \Sigma_i vol(B_i) \leq m \cdot max_i vol(B_i) \leq m/(2-\epsilon)^n$.

This implies no polynomial-time algorithm can estimate the volume to a factor better than exponential in n.

6 PageRank

PageRank is a link analysis algorithm that assigns a numerical weighting to each element of a hyperlinked set of documents, such as the World Wide Web, with the purpose of "measuring" its relative importance within the set. The algorithm may be applied to any collection of entities with reciprocal quotations and references. The numerical weight that it assigns to any given element E is also called the PageRank of E and denoted by $PR[E]$.

$PR[E]$ is a probability distribution used to represent the likelihood that a person randomly clicking on links will arrive at any particular page E. Its computation requires several passes, called "iterations", through the collection to adjust approximate PageRank values to more closely reflect the theoretical true value.

In simplified PageRank algorithm, the initial approximation of PageRank would be evenly divided between the collection of documents. The value of the link-votes is divided among all the outbound links on a page. The PageRank conferred by an outbound link $L()$ is equal to the document's own PageRank score divided by the normalized number of outbound links (it is assumed that links to specific URLs only count once per document).

In PageRank algorithm including damping factor, we assume the probability, at any step, that the person will continue is a damping factor d. The damping factor is subtracted from 1 (and in some variations of the algorithm. The result is divided by the number of documents in the collection). Then it is added to the product of the damping factor and the sum of the incoming PageRank scores.

The formula uses a model of a random surfer who gets bored after several clicks and switches to a random page. The PageRank value of a page reflects the chance that the random surfer will land on that page by clicking on a link. It can be understood as a Markov chain in which the states are pages, and the transitions are all equally probable and are the links between pages.

Let N is the total number of pages and $p_1, p_2, \cdots, p_N$ are the pages under consideration. Then

$$R = \begin{pmatrix} PR[p_1] \\ PR[p_2] \\ \cdots \\ PR[p_N] \end{pmatrix}$$

satisfies

$$\mathbf{R} = \begin{pmatrix} 1-d/N \\ 1-d/N \\ \cdots \\ 1-d/N \end{pmatrix} + \mathbf{d} \begin{pmatrix} l(p_1,p_1) & l(p_1,p_2) & \cdots & l(p_1,p_N) \\ l(p_2,p_1) & l(p_2,p_2) & \cdots & l(p_2,p_N) \\ \cdots & & & \\ l(p_N,p_1) & l(p_N,p_2) & \cdots & l(p_N,p_N) \end{pmatrix} \mathbf{R}$$

where the adjacency function $l(p_i, p_j)$ is 0 if page p_j does not link to p_i, and normalized such that $\forall j, \Sigma_i l(p_i, p_j) = 1$.

7 HITS

Hypertext Induced Topic Selection (HITS) is a link analysis algorithm that rates Web pages for their authority and hub values. Authority value estimates the value of the content of the page; hub value estimates the value of its links to other pages. These values can be used to rank Web search results.

Authority value x and hub value y are defined in terms of one another in a mutual recursion. An authority value is computed as the sum of the scaled hub values that point to that page. A hub value is the sum of the scaled authority values of the pages it points to. So we have $\vec{x} = A^T\vec{y}$ and $\vec{y} = A\vec{x}$. It is equivalent to $\vec{x} = A^TA\vec{x}$ and $\vec{y} = AA^T\vec{y}$.

HITS, like PageRank, is an iterative algorithm based purely on the linkage of the documents on the web. However it does have some major differences:

* It is executed at query time, and not at indexing time, with the associated hit on performance that accompanies query-time processing.

* It is not commonly used by search engines. (Though some sources claim a similar algorithm is used by Ask.com.)

* It computes two scores per document (hub and authority) as opposed to a single score.

* It is processed on a small subset of 'relevant' documents, not all documents as was the case with PageRank.

8 Linear algebra for PageRank and HITS

(Symmetric Shur Decomposition) If $A \in R^{n\times n}$ is symmetric, then there exists a real orthogonal Q such that $Q^TAQ = diag(\lambda_1, \cdots, \lambda_n)$.

Proof: Suppose λ_i is eigenvalue and v_i is the corresponding unit 2-norm eigenvector, then we have $\lambda_i = v_i^HAv_i = v_i^HA^Hv_i = \overline{v_i^HAv_i} = \overline{\lambda_i}$. Therefore $\lambda_i \in R$. So $v_i \in R^n$. Let $Q = (v_1, v_2, \cdots, v_n)$. Then $Q^TAQ = (v_1^T, v_2^T, \cdots, v_n^T)^TA(v_1, v_2, \cdots, v_n)$

$$= \begin{pmatrix} v_1^TAv_1 & v_1^TAv_2 & \cdots & v_1^TAv_n \\ v_2^TAv_1 & v_2^TAv_2 & \cdots & v_2^TAv_n \\ \cdots & & & \\ v_n^TAv_1 & v_n^TAv_2 & \cdots & v_n^TAv_n \end{pmatrix}$$

Because the above matrix is still symmetric, for $i \neq j$ we have $v_i^TAv_j = v_j^TAv_i$. That is $\lambda_jv_i^Tv_j = \lambda_iv_j^Tv_i = \lambda_iv_i^Tv_j$. So if $\lambda_i \neq \lambda_j$, then $v_i^Tv_j = 0$. For eigenvectors that belong to the same eigenvalue, we can use Gram-Schmidt orthogonalization to transform the eigenvectors to the set of mutually orthogonal eigenvectors. So we can pick the set of unit 2-norm eigenvectors $\{v_i\}$ such that its members are mutually orthogonal. Then we have $Q^TAQ = diag(\lambda_1, \cdots, \lambda_n)$.

To solve the linear equation $x = Ax$, we can use the iterative method below:

From arbitrarily picked unit 1-norm x_0,

compute Ax_0, rescale to unit 1-norm x_1,

.

.

.

compute Ax_i, rescale to unit 1-norm x_{i+1},

.

.

.

Suppose $x_0 = \alpha_1 v_1 + \alpha_2 v_2 + \cdots \alpha_n v_n$. Then we have $x_k = \Sigma_i \alpha_i \lambda_i^k v_i / ||\Sigma_i \alpha_i \lambda_i^k v_i||_1$

So $\lim_k x_k$ is convergent. In addition, $\lim_k x_k = A \lim_k x_k$.

9 Buffon's needle

Given a needle of length l dropped on a plane ruled with parallel lines t units apart, what is the probability that the needle will cross a line?

By observation, $E(\#\text{intersections}) = c \cdot l$, for some c regardless of the shape of the needle. To determine c, consider throwing in a circular needle of radius 1. The number of intersections is exactly 2. We thus have $c \cdot 2\pi = 2$. Then $c = 1/\pi$.