0368.416701 Sublinear Time Algorithms October 26, 2015

Lecture 2

Lecturer: Ronitt Rubinfeld *Scribe: Oleg Zlydenko, Nadav Trumer, Gonen Krak*

1 The Element Distinctness Problem

Given n elements $x_1, x_2, ..., x_n$ we wish to determine if the elements are distinct. To solve the problem in a deterministic way, we need to search through the whole input. Therefore, instead of solving this problem, we will solve a simpler property test problem.
Given a set of elements X and an error parameter ε:

- If all the elements of X are distinct output $PASS$
- If more than εn duplicates (i.e. X is far from distinct) output FAIL with probability $\geq \frac{3}{4}$
- otherwise any answer is OK

Note that because this problem is *symmetric*, i.e. the order of the elements does not affect the tested property, it's not hard to see that we can't do any better than sample some number of elements and check if there are any duplicates. Therefore, the interesting question is how many samples we will use. The birthday paradox says that in general will need at least $O(\sqrt{n})$ samples, but will it be sufficient?

1.1 Algorithm Outline

The basic algorithm to solve this problem is indeed sampling $c \cdot \frac{\sqrt{n}}{\varepsilon}$ samples and testing if there are any duplicates among them. For the sake of simpler analysis, we will devise a modified algorithm that works in two stages:

1. In the first stage, we will find the first element in many ordered duplicate pairs
2. In the second stage, we will find the second element in those pairs for one of the duplicate pairs whose first element we found in stage one.

We will use the following algorithm:

Algorithm 1 Testing Element Distinctness

1: Sample $\sqrt{n}$ samples from X, call them $S1$
2: Test for duplicates in $S1$. If found, output FAIL
3: Sample $\frac{c\sqrt{n}}{\varepsilon}$ samples, call them $S2$
4: Test for each element in $S2$ if we already found him in $S1$
5: If we found such a pair, output FAIL
6: Otherwise output PASS

Remark We can use hashing to check for duplicates and therefore the running time of the algorithm is $O(\frac{\sqrt{n}}{\varepsilon})$

Remark Note that if Algorithm 1 solves the property testing problem then the basic algorithm also solves it, since Algorithm 1 uses at most $O(\frac{\sqrt{n}}{\varepsilon})$ samples

1.2 Algorithm Analysis

Clearly, if all the elements in X are distinct the algorithm always outputs PASS. We need to show that if there are more than εn duplicates the algorithm outputs FAIL with high probability (w.h.p).

1.2.1 Pair duplicate elements of X

For the sake of analysis, we divide the elements of X into the set P of its ordered pairs of duplicate elements. It is important to note that elements without duplicates do not get paired and if there is an odd number of duplicates, the last one does not get paired.
For example, if $X = \{7, 1, 2, 1, 3, 4, 2, 1, 2, 4, 4, 3, 5, 3, 6, 4\}$ then the elements that will get paired are $\{1, 1, 1\}$, $\{2, 2, 2\}$, $\{3, 3, 3\}$ and $\{4, 4, 4, 4\}$. Then we will have $P = \{(1, 1), (2, 2), (3, 3), (4, 4), (4, 4)\}$.

Claim 1 *If there are many duplicates, i.e. the number of duplicates is at least εn then $|P| \geq \frac{\varepsilon n}{3}$.*

Note that if there are even number of duplicates(of each element) then we simply have $|P| \geq \frac{\varepsilon n}{2}$. If there is an odd number of duplicates for some element, then all of its duplicates except for one will be paired. Therefore if the element appears k times, we will have $\frac{k-1}{2} \geq \frac{k}{3}$ such pairs of this element, and since k is at least 2 and odd, we have $k \geq 3$ and overall we get $|P| \geq \frac{\varepsilon n}{3}$.

1.2.2 Many First Elements of Pairs in P Get Hit (whp)

We show that in the first stage of the algorithm we hit the first element of an ordered pair w.h.p.
Let $F = \{i | (i, j) \in P \wedge i \in S_1\}$. The probability that one sample k from S_1 hits some first index of an ordered pair of P is Pr[sample $k \in S_1$ hits 1st index in P]$= \frac{|P|}{n} \geq \frac{\frac{\varepsilon n}{3}}{n} = \frac{\varepsilon}{3}$. For each k define an indicator y_k that gets 1 if sample k hits some first index in P and 0 otherwise. We have

$$E[y_k] = Pr[y_k = 1] \geq \frac{\varepsilon}{3}$$

Denote $Y = \sum_{k \in S_1} y_k$. Then,

$$E[Y] = \sum_{k \in S_1} E[y_k] \geq \frac{\varepsilon\sqrt{n}}{3}$$

We need to note two things: First, $E[Y] \neq E[|F|]$ since Y counts repetitions as well (if the first same index in P was hit two times we will count it as two in P and as one in F). Second, we care about the probability that $|F|$ is high and not in its expected value. We will overcome those obstacles using the following observations:
From the Chernoff bound we have

$$Pr[Y < \frac{E[Y]}{2}] < e^{-(\frac{1}{8} \cdot \frac{\varepsilon}{3}\sqrt{n})} << \frac{1}{12}$$

for large enough n.
Let's call $Y < \frac{E[Y]}{2}$ bad event 1. We showed that this bad event happens with low probability.
Now, define an indicator $Z_{k,l}$ that gets 1 if samples k, l hit the same element(same index) and 0 otherwise. Denote $Z = \sum_{k \neq l \in S_1} Z_{k,l}$. It is easy to see that Z is an upper bound on the repetitions in Y, and in fact a very crude one, since if a pair was hit m times it will be count $\binom{m}{2}$ times in Z. Therefore we have $F \geq Y - Z$. We show that w.h.p Z is small:

$$E[Z_{k,l}] = Pr[Z_{k,l} = 1] = \frac{1}{n}$$

and

$$E[Z] = \sum_{k \neq l \in S_1} E[Z_{k,l}] = \binom{\sqrt{n}}{2} \cdot \frac{1}{n} < \frac{1}{2}$$

For Markov's inequality we get $Pr[Z > 6] < \frac{1}{12}$. Let the event $Z > 6$ be bad event 2.

Therefore, with probability $\geq \frac{5}{6}$ bad events 1 and 2 do not occur and we have

$$|F| \geq Y - Z \geq \frac{\varepsilon\sqrt{n}}{6} - 6 \geq \frac{\varepsilon\sqrt{n}}{12}$$

1.2.3 Second element of a pair is hit w.h.p

We showed that w.h.p stage one succeeds and we have $|F| \geq \frac{\varepsilon\sqrt{n}}{12}$. The probability that sample k from S_2 hits the second element of one of the indexes in F is
Pr[sample $k \in S_2$ hits 2nd number of some pair in F]$= \frac{|F|}{n} \geq \frac{\varepsilon}{12\sqrt{n}}$. Therefore, the probability of failure of the algorithm, i.e. the probability that all $k \in S_2$ missed all the second elements is

$$Pr[failure] \leq (1 - \frac{\varepsilon}{12\sqrt{n}})^{\frac{c\sqrt{n}}{\varepsilon}} = e^{-c'} \leq \frac{1}{12}$$

where c and c' are chosen accordingly. We say that the event that none of the elements in F was hit by a sample from S_2 is bad event 3. We showed that the algorithm succeeds if all of the bad events do not occur. The probability that one of them occurs is at most $3 \cdot \frac{1}{12} = \frac{1}{4}$ and therefore the algorithm succeeds with probability $\geq \frac{3}{4}$.

2 The Uniformity Testing Problem

Let P be an unknown distribution over some known domain D with $|D| = n$.
Since P is unknown, we can't determine P in sub-linear time, but in many times it can be beneficial to understand some properties about P - whether P is uniform, monotonic, or estimate its entropy.
In this section we will discuss that uniformity testing problem
We would like to devise an algorithm such that given an unknown distribution P:

- If $P = U_D$ where U_D is the uniform distribution over D output PASS (with prob. $\geq \frac{3}{4}$)
- If $||P - U_D|| \geq \varepsilon$ output FAIL (with prob. $\geq \frac{3}{4}$)

There are many possible candidates for the distance function between distributions.
We will focus on two of them:

1. $L_1 : ||P - Q||_1 = \sum_{x \in D} |p(x) - q(x)|$
2. $L_2 : ||P - Q||_2 = \sqrt{\sum_{x \in D} (p(x) - q(x))^2}$

Claim 2 $||P - Q||_2 \leq ||P - Q||_1 \leq \sqrt{n} \cdot ||P - Q||_2$

2.1 Linear Time "Plug-in" Algorithm for Uniformity Testing

We will show a simple linear-time "Plug in" algorithm for testing uniformity with the $L1$ norm

Algorithm 2 Plug-in Estimate

1: Sample $m = \frac{n}{\varepsilon^2} \cdot \log n$ samples from P
2: for every $x \in D$ we evaluate $\hat{p}(x) = \frac{\#x}{m}$ where $\#x$ is the number of times we sampled the value x
3: If $\sum_{x \in D} |\hat{p}(x) - \frac{1}{n}| > \frac{\varepsilon}{2}$ output FAIL
4: Otherwise output PASS

It is left as an exercise to show that if we take $m = \frac{n}{\varepsilon^2} \cdot \log n$ samples from P then with probability $\geq \frac{99}{100}$ we have $\sum_{x \in D} |\hat{p}(x) - p(x)| \leq \frac{\varepsilon}{2}$.
We will now prove the correctness of the algorithm:
If $P = U_D$ then $\sum |\hat{p}(x) - p(x)| = \sum |\hat{p}(x) - \frac{1}{n}| < \frac{\varepsilon}{2}$ so we output PASS w.h.p.
On the other side assume $||P - U_D||_1 > \varepsilon$. Assume that the algorithm outputs PASS, i.e. $||\hat{p} - U_D|| \leq \frac{\varepsilon}{2}$. From the triangle inequality,

$$||P - U_D||_1 = ||P - \hat{P} + \hat{P} - U_D||_1 \leq ||P - \hat{P}||_1 + ||\hat{P} - U_D||_1 \leq \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon$$

which is a contradiction. Therefore, if $||P - U_D||_1 > \varepsilon$, the algorithm will output FAIL w.h.p.