Sublinear Algorithms
Lecture 1

Sofya Raskhodnikova
Boston University
Organizational

Course webpage:
https://cs-people.bu.edu/sofya/sublinear-course/

Use Piazza to ask questions
Office hours (on zoom):
**Wednesdays**, 1:00PM-2:30PM

Evaluation
• Homework (about 4 assignments)
• Taking lecture notes (about once per person)
• Course project and presentation
• Peer grading (PhD student only)
• Class participation
Tentative Topics

Introduction, examples and general techniques.

Sublinear-time algorithms for
- graphs
- strings
- geometric properties of images
- basic properties of functions
- algebraic properties and codes
- metric spaces
- distributions
  Tools: probability, Fourier analysis, combinatorics, codes, …

Sublinear-space algorithms: streaming
Tentative Plan

Introduction, examples and general techniques.


Lecture 2. (Next week) Properties of functions and graphs. Sublinear approximation.

Lecture 3-5. Background in probability. Techniques for proving hardness. Other models for sublinear computation.
Motivation for Sublinear-Time Algorithms

Massive datasets
- world-wide web
- online social networks
- genome project
- sales logs
- census data
- high-resolution images
- scientific measurements

Long access time
- communication bottleneck (slow connection)
- implicit data (an experiment per data point)
Do We Have To Read All the Data?

- What can an algorithm compute if it
  - reads only a **tiny** portion of the data?
  - runs in **sublinear** time?

Image source: http://apandre.wordpress.com/2011/01/16/bigdata/
A Sublinear-Time Algorithm

randomized algorithm

approximate answer

Quality of approximation

Resources
- number of queries
- running time
Goal: Fundamental Understanding of Sublinear Computation

- What computational tasks?
- How to measure quality of approximation?
- What type of access to the input?
- Can we make our computations robust (e.g., to noise or erased data)?
Types of Approximation

Classical approximation
• need to compute a value
  ➢ output should be close to the desired value
  ➢ example: average

Property testing
• need to answer YES or NO
  ➢ Intuition: only require correct answers on two sets of instances that are very different from each other
Classical Approximation

A Simple Example
**Approximate Diameter of a Point Set** [Indyk]

**Input:** $m$ points, described by a distance matrix $D$
- $D_{ij}$ is the distance between points $i$ and $j$
- $D$ satisfies triangle inequality and symmetry
  (Note: input size is $n = m^2$)

- Let $i, j$ be indices that maximize $D_{ij}$.
- Maximum $D_{ij}$ is the *diameter*.

**Output:** $(k, \ell)$ such that $D_{k\ell} \geq D_{ij} / 2$
Algorithm and Analysis

Algorithm \((m,D)\)

1. Pick \(k\) arbitrarily
2. Pick \(\ell\) to maximize \(D_{k\ell}\)
3. Output \((k, \ell)\)

**Approximation guarantee**

\[
D_{ij} \leq D_{ik} + D_{kj} \quad \text{(triangle inequality)}
\]

\[
\leq D_{k\ell} + D_{k\ell} \quad \text{(choice of } \ell + \text{symmetry of } D)
\]

\[
\leq 2D_{k\ell}
\]

**Running time:** \(O(m) = O(m = \sqrt{n})\)

A rare example of a **deterministic** sublinear-time algorithm
Property Testing
Property Testing: YES/NO Questions

Does the input satisfy some property? (YES/NO)

“in the ballpark” vs. “out of the ballpark”

Does the input satisfy the property or is it far from satisfying it?

- for some applications, it is the right question (probabilistically checkable proofs (PCPs), precursor to learning)
- good enough when the data is constantly changing
- fast sanity check to rule out inappropriate inputs (rejection-based image processing)
Property Tester Definition

Probabilistic Algorithm

- **YES:** Accept with probability $\geq \frac{2}{3}$
- **NO:** Reject with probability $\geq \frac{2}{3}$

Property Tester

- **YES:** Accept with probability $\geq \frac{2}{3}$
- **Far from YES:** Don’t care
- **Reject with probability $\geq \frac{2}{3}$

$\varepsilon$-far = differs in many places ($\geq \varepsilon$ fraction of places)
Randomized Sublinear Algorithms

Toy Examples
Property Testing: a Toy Example

Input: a string \( w \in \{0,1\}^n \)

Question: Is \( w = 00 \ldots 0 \)?

Requires reading entire input.

Approximate version: Is \( w = 00 \ldots 0 \) or does it have \( \geq \varepsilon n \) 1’s (“errors”)?

<table>
<thead>
<tr>
<th>Test ((n, w))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Sample ( s = 2/\varepsilon ) positions uniformly and independently at random</td>
</tr>
<tr>
<td>2. If 1 is found, reject; otherwise, accept</td>
</tr>
</tbody>
</table>

Analysis: If \( w = 00 \ldots 0 \), it is always accepted.

If \( w \) is \( \varepsilon \)-far, \( \Pr[\text{error}] = \Pr[\text{no 1's in the sample}] \leq (1 - \varepsilon)^s \leq e^{-\varepsilon s} = e^{-2} < \frac{1}{3} \)

Witness Lemma

If a test catches a witness with probability \( \geq p \), then \( s = \frac{2}{p} \) iterations of the test catch a witness with probability \( \geq 2/3 \).
Input: a string \( w \in \{0,1\}^n \)

Goal: Estimate the fraction of 1’s in \( w \) (like in polls)

It suffices to sample \( s = \frac{1}{\epsilon^2} \) positions and output the average to get the fraction of 1’s \( \pm \epsilon \) (i.e., additive error \( \epsilon \)) with probability \( \geq \frac{2}{3} \)

**Hoeffding Bound**

Let \( Y_1, \ldots, Y_s \) be independently distributed random variables in \([0,1]\).

Let \( Y = \frac{1}{s} \sum_{i=1}^{s} Y_i \) (called sample mean). Then \( \Pr[|Y - E[Y]| \geq \epsilon] \leq 2e^{-2s\epsilon^2} \).

\( Y_i = \text{value of sample } i \). Then \( E[Y] = \frac{1}{s} \sum_{i=1}^{s} E[Y_i] = \text{(fraction of 1's in } w) \)

\[ \Pr[(\text{sample mean}) - \text{(fraction of 1's in } w)] \geq \epsilon \leq 2e^{-2s\epsilon^2} = 2e^{-2} < 1/3 \]

Apply Hoeffding Bound substitute \( s = \frac{1}{\epsilon^2} \)
Property Testing

Simple Examples
Testing Properties of Images
Pixel Model

Input: $n \times n$ matrix of pixels
(0/1 values for black-and-white pictures)

Query: point $(i_1, i_2)$

Answer: color of $(i_1, i_2)$
Testing if an Image is a Half-plane [R03]

A half-plane or $\varepsilon$-far from a half-plane?

$O(1/\varepsilon)$ time
Half-plane Instances

A half-plane

\( \frac{1}{4} \)-far from a half-plane
Half-plane Instances

A half-plane

\( \frac{1}{4} \)-far from a half-plane
Half-plane Instances

A half-plane

\[ \frac{1}{4}\text{-far from a half-plane} \]
Half-plane Instances

A half-plane

$1/4$-far from a half-plane
Half-plane Instances

A half-plane

$\frac{1}{4}$-far from a half-plane
Half-plane Instances

A half-plane

\[ \frac{1}{4} \text{-far from a half-plane} \]
**Half-plane Instances**

A half-plane

$\frac{1}{4}$-far from a half-plane
Strategy

“Testing by implicit learning” paradigm

- Learn the outline of the image by querying a few pixels.
- Test if the image conforms to the outline by random sampling, and reject if something is wrong.
Half-plane Test

Claim. The number of sides with different corners is 0, 2, or 4.

Algorithm
1. Query the corners.
Half-plane Test: 4 Bi-colored Sides

Claim. The number of sides with different corners is 0, 2, or 4.

Analysis

- If it is 4, the image cannot be a half-plane.

Algorithm

1. Query the corners.
2. If the number of sides with different corners is 4, reject.
Claim. The number of sides with different corners is 0, 2, or 4.

Analysis

- If all corners have the same color, the image is a half-plane if and only if it is unicolored.

Algorithm

1. Query the corners.
2. If all corners have the same color $c$, test if all pixels have color $c$ (as in Toy Example 1).
Half-plane Test: 2 Bi-colored Sides

Claim. The number of sides with different corners is 0, 2, or 4.

Analysis

1. The area outside of $W \cup B$ has $\leq \frac{\varepsilon n^2}{2}$ pixels.
2. If the image is a half-plane, $W$ contains only white pixels and $B$ contains only black pixels.
3. If the image is $\varepsilon$-far from half-planes, it has $\geq \frac{\varepsilon n^2}{2}$ wrong pixels in $W \cup B$.
4. By Witness Lemma, $\frac{4}{\varepsilon}$ samples suffice to catch a wrong pixel.

Algorithm

1. Query the corners.
2. If # of sides with different corners is 2, on both sides find 2 different pixels within distance $\varepsilon n/2$ by binary search.
3. Query $\frac{4}{\varepsilon}$ pixels from $W \cup B$
4. Accept iff all $W$ pixels are white and all $B$ pixels are black.
Testing if an Image is a Half-plane \[^{[R03]}\]

A half-plane or \(\varepsilon\)-far from a half-plane?

\[O(1/\varepsilon)\] time \(\checkmark\)
Other Results on Testing Properties of Images

• Pixel Model

**Convexity** [Berman Murzabulatov R]
Convex or $\varepsilon$-far from convex?

$$O(1/\varepsilon) \text{ time}$$

**Connectedness** [Berman Murzabulatov R]
Connected or $\varepsilon$-far from connected?

$$O\left(1/\varepsilon^{3/2} \sqrt{\log 1/\varepsilon}\right) \text{ time}$$

**Partitioning** [Kleiner Keren Newman 10]
Can be partitioned according to a template
or is $\varepsilon$-far?

time independent of image size

• Properties of sparse images [Ron Tsur 10]