Lecture 17: Sublinear-time algorithms

COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

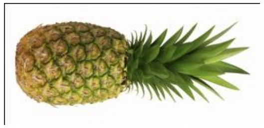
Administrivia, Plan

- Admin:
  - My office hours after class (CSB517)
- Plan:
  - Finalize embeddings
  - Sublinear-time algorithms
  - Projects
- Scriber?

Embeddings of various metrics into ℓ_1

Metric	Upper bound
Earth-mover distance (s-sized sets in 2D plane)	$O(\log s)$ [Cha02, IT03]
Earth-mover distance (s-sized sets in $\{0,1\}^d$)	$O(\log s \cdot \log d)$ [AIK08]
Edit distance over $\{0,1\}^d$ (#indels to transform x->y)	$2^{\tilde{O}(\sqrt{\log d})}$ [OR05]
Ulam (edit distance between permutations)	$O(\log d)$ [CK06]
Block edit distance	$\tilde{O}(\log d)$ [MS00, CM07]

edit( ,

 = 2

edit(1234567,
7123456) = 2

Non-embeddability into ℓ_1

Distortion D implies sketch (decision version) with $O(D)$ approximation and $O(1)$ size! (implies NNS)

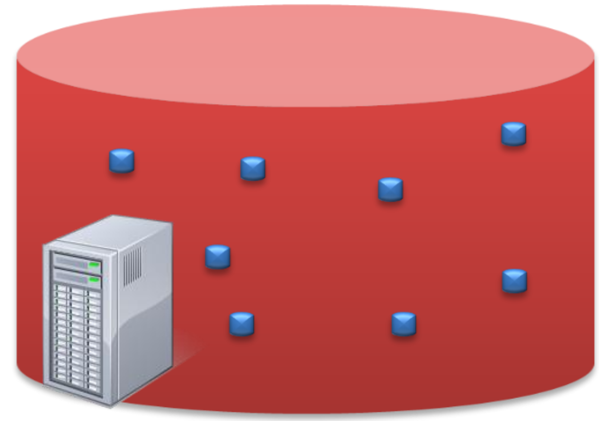
OPEN to get better for pretty much all these distances!

Earth-mover distance (s-sized sets in 2D plane)	$O(\log s)$ [Cha02, IT03]	$\Omega(\sqrt{\log s})$ [NS07]
Earth-mover distance (s-sized sets in $\{0,1\}^d$)	$O(\log s \cdot \log d)$ [AIK08]	$\Omega(\log s)$ [KN05]
Edit distance over $\{0,1\}^d$ (#indels to transform x->y)	$2^{\tilde{O}(\sqrt{\log d})}$ [OR05]	$\Omega(\log d)$ [KN05,KR06]
Ulam (edit distance between permutations)	$O(\log d)$ [CK06]	$\tilde{\Omega}(\log d)$ [AK07]
Block edit distance	$\tilde{O}(\log d)$ [MS00, CM07]	4/3 [Cor03]

Sublinear-time algorithms

Setup

- Can we get away with not even looking at all data?
  - Just use a sample…
- Where do we get samples?
  - stored on disk, passing through a router, etc
  - Data comes as a sample
  - Observation of a "natural" phenomenon



Two types of algorithms

- Classic:
  - Output an answer, approximately
  - E.g.: number of triangles in a graph!

- Property testing:
  - Does object O have *blah* property or not
  - E.g.: does graph have a triangle or not
  - Distribution testing: O=distribution
  - Need a new notion of approximation

Distribution Testing

- Problem:
  - given m samples $x_1, \dots x_m \in \{1,2,\dots n\}$, from D
  - do they come from a *uniform distribution*?
- Hard to solve precisely:
  - Uniform except 6 has probability 10^{-50} higher than normal
  - Do we care about 10^{-50}?
- Use approximation…

Approximation: total variation

- Goal: distinguish between
  - exactly uniform
  - *sufficiently non-uniform*:
    - ε-far: $||D - U_n||_1 \geq \epsilon$
- Why ℓ_1 distance?
  - Equivalent to Total Variation distance:
  - How to distinguish distributions A, B with 1 sample?
    - a test: is a set $T \subset [n]$
    - Check whether a sample $x \in T$
    - Distinguishing probability: $\left|\Pr_A[x \in T] - \Pr_B[x \in T]\right|$
    - We want the best such test:

$$TV(A, B) = \max_{T \subset [n]} \left|\Pr_A[x \in T] - \Pr_B[x \in T]\right|$$

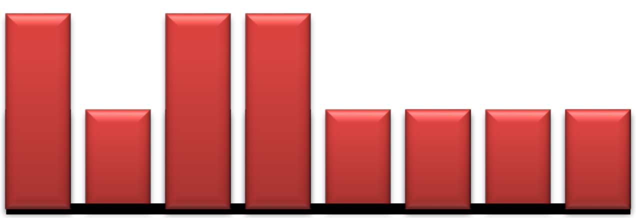
  - Claim: $TV(A, B) = \frac{1}{2}||A - B||_1$
- $||D - U_n||_1 \leq \epsilon$ means:
  - sampling up to ~1/ε times nearly-equivalent to sampling from a uniform distribution

Algorithm attempt

- How shall we test uniformity?
  - Estimate distribution empirically, $\widehat{D}$
  - Compute $||\ \widehat{D} - U_n||$...
  - How many samples do we need?
    - At least $n/2$: if half the coordinates are zero, far from uniform!
  - χ^2 test: also $\Omega(n)$ samples
- Can we do better?
- Theorem: can test uniformity with $O_\epsilon(\sqrt{n})$ samples

Algorithm for Uniformity

- Counts the number of collisions
- Intuition:
  - If not uniform, more likely to have more collisions



Algorithm UNIFORM:

Input: $n, m, x_1, \ldots x_m$
 C = 0;
 for(i=0; i<m; i++)
  for(j=i+1; j<m; j++)
   if $(x_i = x_j)$
    C++;

 if $(C < a \cdot m^2/n)$
  return “Uniform”;
 else
  return “Not uniform”;
// a: constant dependent on ε

Algorithm intuition

- Uses $\sim\sqrt{n}$ samples
  - as long as all distinct, no way to tell apart
  - first collisions appear at $\sim\sqrt{n}$ - the birthday paradox!

```
Algorithm UNIFORM:

Input: n, m, x_1, … x_m
 C = 0;
 for(i=0; i<m; i++)
  for(j=i+1; j<m; j++)
   if (x_i = x_j)
    C++;

 if (C < a · m^2/n)
  return "Uniform";
 else
  return "Not uniform";
// a: constant dependent on ε
```

Analysis

- Consider ℓ_2 distance!
- If $D = U_n$
  - $||D - U_n||_2 = 0$
- If $||D - U_n||_1 \geq \epsilon$
  - $||D - U_n||_2^2 > \epsilon^2/n$
- Claim:

  $||D - U_n||_2^2 = ||D||_2^2 - 1/n$
- Hence, enough to distinguish:
  - $||D||_2^2 = 1/n$ (unif)
  - $||D||_2^2 > 1/n + \epsilon^2/n$ (non-unif)
- Compute $||D||_2^2$ up to additive ϵ^2/n ?

Algorithm UNIFORM:

Input: $n, m, x_1, \dots x_m$

```
  C = 0;
  for(i=0; i<m; i++)
    for(j=i+1; j<m; j++)
      if (x_i = x_j)
        C++;

  if (C < a · m^2/n)
    return "Uniform";
  else
    return "Not uniform";
// a: constant dependent on ε
```

Analysis

- New goal: distinguish
  - $||D||_2^2 = 1/n$
  - $||D||_2^2 > 1/n + \epsilon^2/n$
- Lemma: $\frac{1}{M} \cdot$[# collisions] is a good enough as long as
  - $m = \Omega\left(\frac{\sqrt{n}}{\epsilon^4}\right)$
  - where $M = m(m-1)/2$

Algorithm UNIFORM:

```
Input: n, m, x_1, ... x_m
 C = 0;
 for(i=0; i<m; i++)
  for(j=i+1; j<m; j++)
   if (x_i = x_j)
    C++;

 if (C < a · m^2/n)
  return "Uniform";
 else
  return "Not uniform";
// a: constant dependent on ε
```

- Projects