Fast, Small-Space Algorithms for Approximate Histogram Maintenance (on a Stream).

A. Gilbert, S. Guha, P. Indyk,

Y. Kotidis, S. Muthukrishnan,

M. Strauss

A data stream

- Data items/updates arrive one at a time
- Small storage, no random access to data unless stored

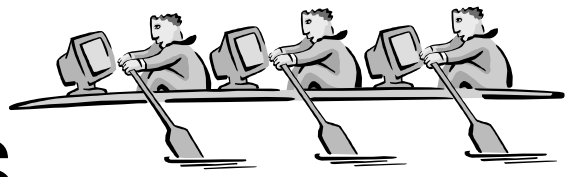
Dimensionality reduction

- Johnson-Lindenstrauss Lemma:
  - x is an n-dimensional vector
  - A is a random n times k matrix, each entry independently drawn from e.g. Gaussian distribution, $k=O(\log N/\varepsilon^2)$
  - Then with probability 1-1/N

$$\|x\|_2 \leq \|Ax\|_2 \leq (1+\varepsilon)\|x\|_2$$

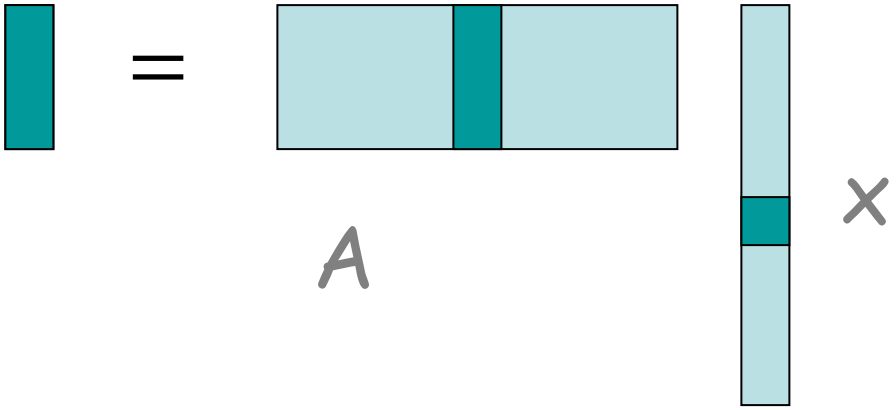
- A can be pseudo-random

What it means



- Can maintain the sketch Ax of x when the coordinates are incremented:

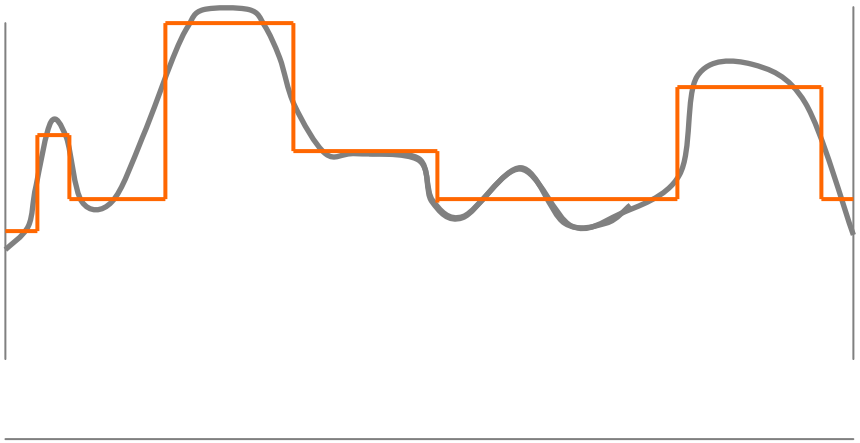
$$A(x+b)=Ax+Ab$$

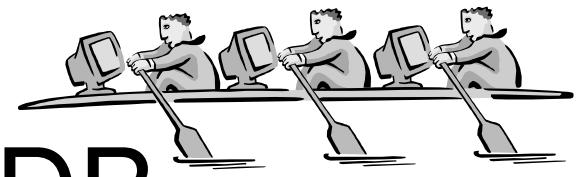


- Can maintain approximate 2-norm of x

Histograms

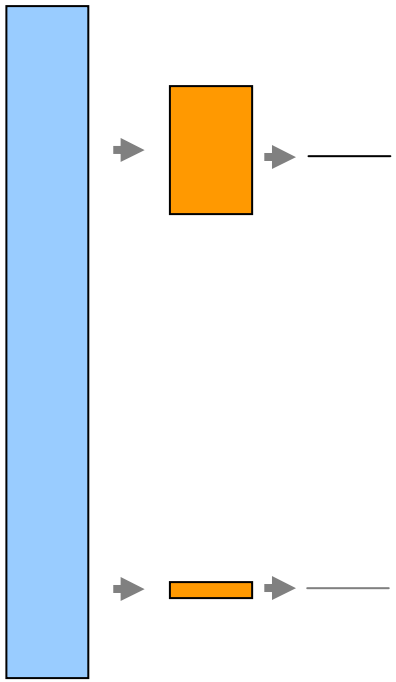
- View x as a function x:[1…n] -> [1…M]
- Approximate it using piecewise constant function h, with B pieces (buckets)





Example app in DB

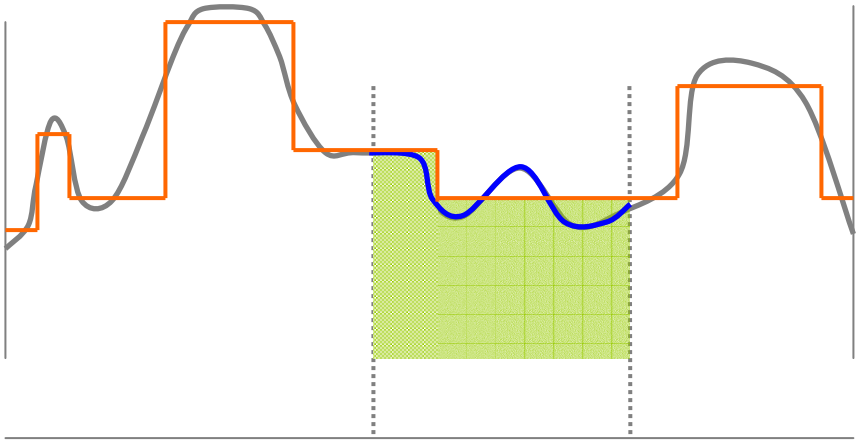
- Find all Indians worth $200K - $300K



1. Select on country
2. Select on worth

1. Select on worth
2. Select on country

Example app continued



Our goal

- Want to maintain the best B-bucket representation of x, under changes of x
- Measure the error using 2-norm (1-norm also OK)

Our Approach

- Maintain sketches Ax of x
- Using Ax, construct B-histogram h which approximately minimizes ||x-h||

Our result

Can maintain a B-histogram h which minimizes ||x-h|| up to a factor of $(1+\varepsilon)$, using poly(log n, B, $1/\varepsilon$) time/space, with probability 1-1/poly(n)

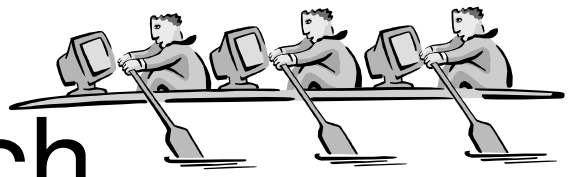
Proof: by iterated improvement

- B buckets, $>n^B$ construction time
- B log n buckets, n^3 construction time
- B $\log^2 n$ buckets, n^2 construction time
- B $\log^2 n$ buckets, n poly(B+log n) time
- B $\log^{O(1)}$ n buckets, poly(B+log n) time
- B buckets, poly(B+log n) time

Exponential time approach

- There are at most $(Mn^2)^B$ functions h
- By JL lemma, can reduce dimension to O(B log n), and approximately preserve ||x-h|| for all h
- To reconstruct h, minimize ||Ax-Ah||
- Can be trivially done by enumerating all h's

Greedy approach



- Start from h=0
- Let χ_I be the characteristic function over interval I
- Find c and I minimizing

$$\|Ax - A(h \oplus c\chi_I)\|_2$$

- $h \leftarrow h \oplus c\chi_I$ & repeat

Details

- The square of

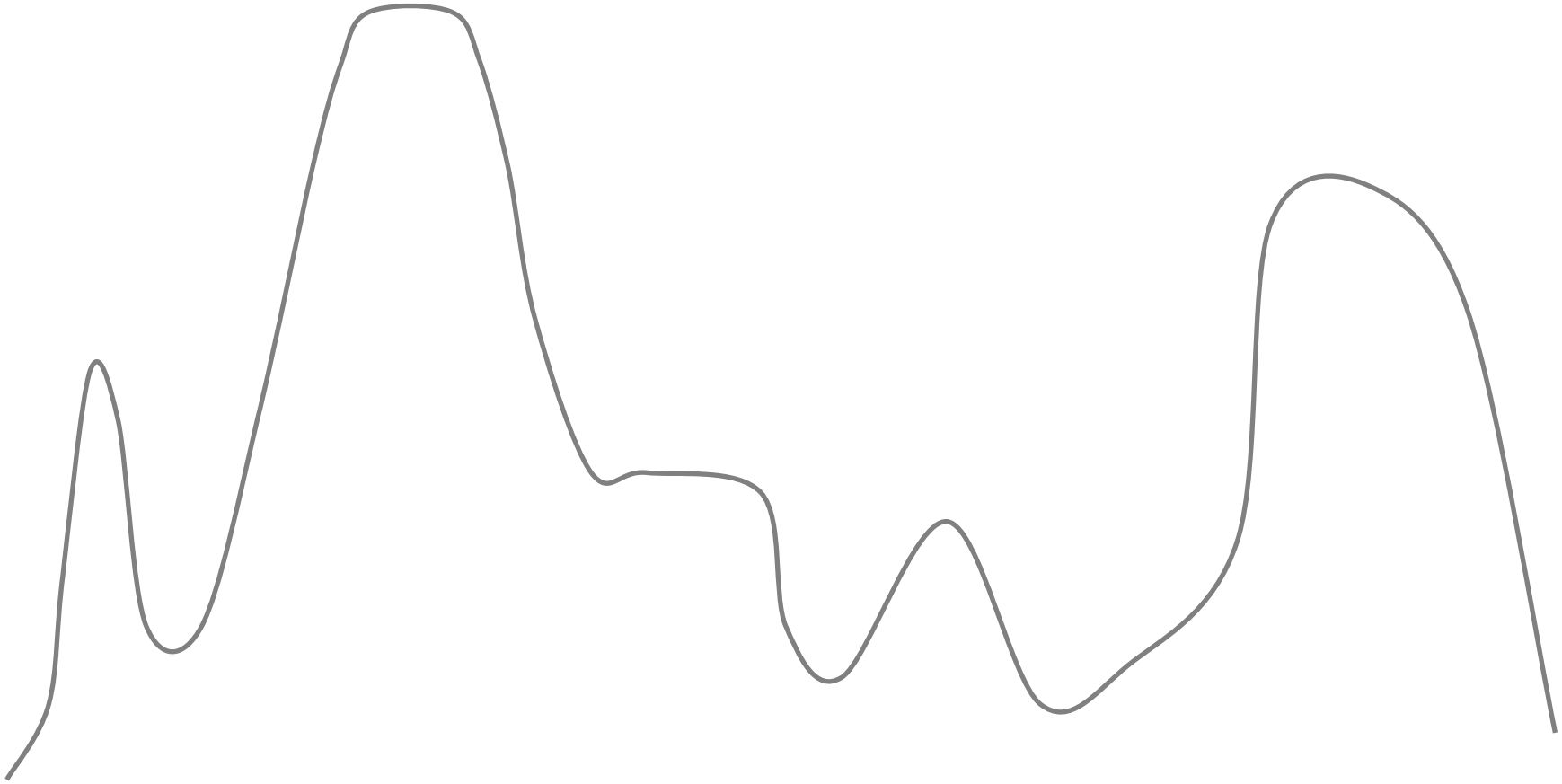
$$\left\| \boldsymbol{Ax} - \boldsymbol{A}(\mathbf{h} \oplus \mathbf{c}\chi_{\mathbf{I}}) \right\|_2$$

  is a quadratic function of c

- Once we compute the parameters of this function, e.g. $E(c)=Ac^2+Bc+D$,

  the minimum is achieved for $c=B/(2A)$

Example



How does it help

- $O(n^2)$ intervals
- $O(n)$ time to find best c minimizing

$$\left\| Ax - A(h \oplus c\chi_I) \right\|_2$$

- Overall: $O(n^3)$ time, $O(k \log (nM))$ intervals

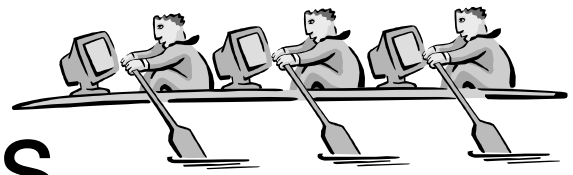
Approximation factor

- Assume $\varepsilon=0$, for simplicity
- Let h* be the optimal k-histogram
- If we replaced the current histogram h by all k intervals of h* (with proper values c), we would reduce the squared error from $||x-h||^2$ to $||x-h^*||^2$
- Thus, there is an interval I of h* (and c) such that

$$||x-h||^2-||x - h \oplus c\chi_I||^2 > 1/k \ (||x-h||^2 -||x-h^*||^2)$$

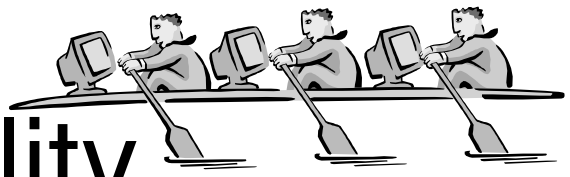
- $O(k \log (nM^2))$ intervals enough to reduce the error to about $||x-h^*||^2$

Dyadic intervals



- Each interval can be decomposed into log n dyadic intervals [1,1],[2,2]…[1,2]...[1,4]
- We can assume opt h is defined by B log n dyadic intervals
- The number of dyadic intervals is n log n
- Reduces the time to n^2 log n

Range summability



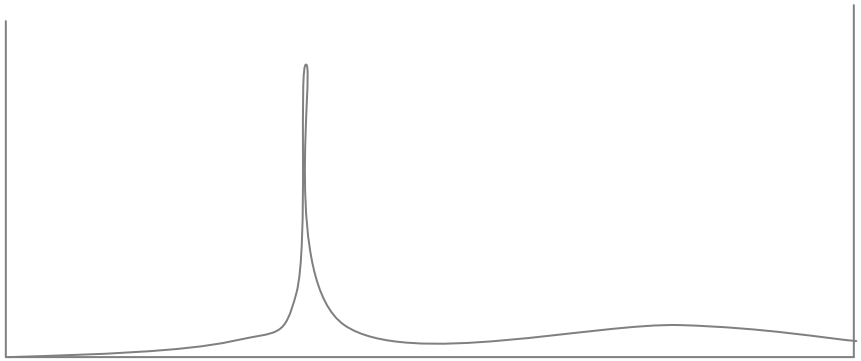
- Recall $\|Ax - A(h \oplus c\chi_I)\|_2$
- Need to compute $A \bullet \chi_I$ i.e., range sum of random variables
- Goal: time polylog n

Naor & Reingold construction

- Method:
  - Generate sum of $a_1, a_2, \ldots, a_n$
  - Generate sum of left half, conditioned on the total sum
  - Recurse
- Conditional distributions are explicit
- The generation can be simulated by Nisan's PRG
- Result: reduces the time to n polylog n

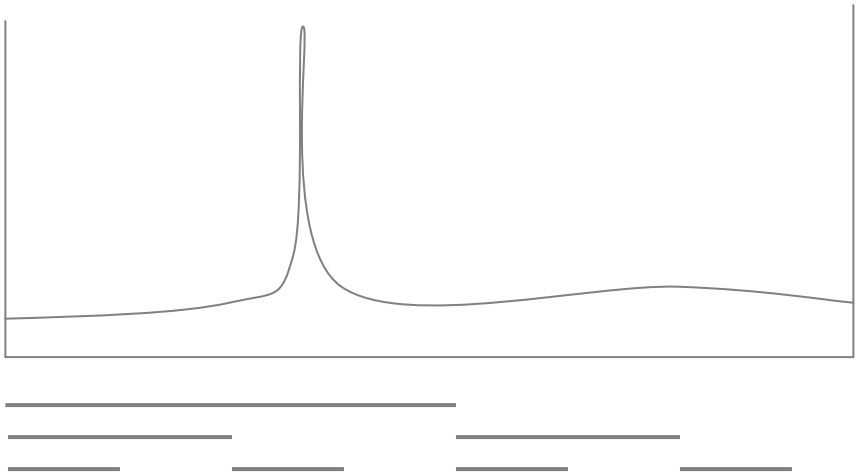
Fast selection of good intervals

- Find which (dyadic) intervals to add in polylog n time
- Consider interval of length 1
- Need to find a “spike” in h-x (if exists)
- Assume only one spike



Chasing Bits

- Non-adaptive binary search



- Essentially, we compose the signal with a filter

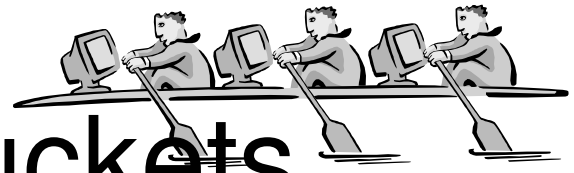
More spikes

- There are few large spikes
- Permute coordinates using pair-wise independent permutation.
- Likely that each interval contains only one spike
- Caveat : how does it work with the range summability
- Result: reduces the time to polylog n

Where are we

- We managed to reduce the time to polylog n
- However, the number of buckets is B polylog n
- Need to reduce the number of buckets to B

Getting rid of the buckets



- B buckets, but O(1)-approximation:
  - Compute h with B polylog n buckets
  - Find h' with B buckets closest to h
    - An off-line problem
    - Can be done approximately using dynamic programming
  - Factor O(1) by triangle inequality
- Factor $(1+\varepsilon)$ is a mess (esp. for 1-norm)

Conclusions

- Can efficiently maintain compact representation of an array of numbers under additive changes
- Works well in practice [TGIK'02]