Trusting the Cloud with Practical Interactive Proofs

Graham Cormode

G.Cormode@warwick.ac.uk

Amit Chakrabarti (Dartmouth)

Andrew McGregor (U Mass Amherst)

Justin Thaler (Harvard/Yahoo!)

Suresh Venkatasubramanian (Utah)

There are no guarantees in life

- From the terms of service of a certain cloud computing service...
- Can we obtain guarantees of correctness of the computation?
  - Without repeating the computation?
  - Without storing all the input?

YES!

Interactive Proofs

What's the answer?

42

Prove it!

10101010010001101101011000100001

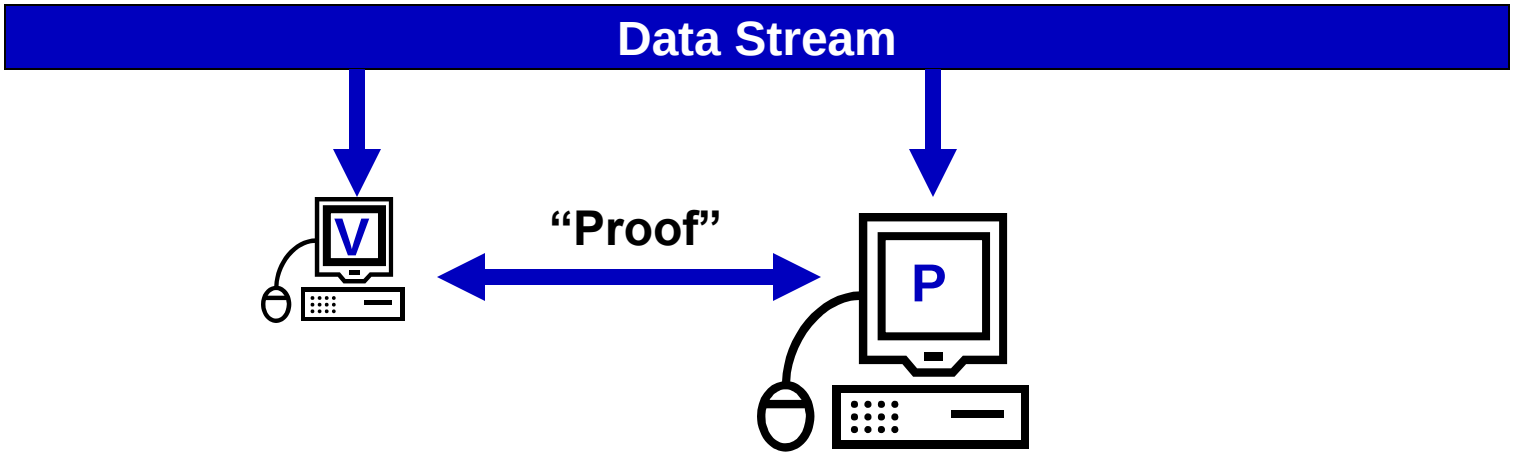
110101?

11010010001000110101010010001101

OK!

(Streaming) Interactive Proofs

- Two party-model: outsource to a more powerful "prover"
  - Fundamental problem: how to be sure that the prover is honest?
- Prover provides "proof" of the correct answer
  - Ensure that "verifier" has very low probability of being fooled
  - Measure resources of the participants, rounds of interaction
  - Related to communication complexity Arthur-Merlin model, and Algebrization, with additional streaming constraints

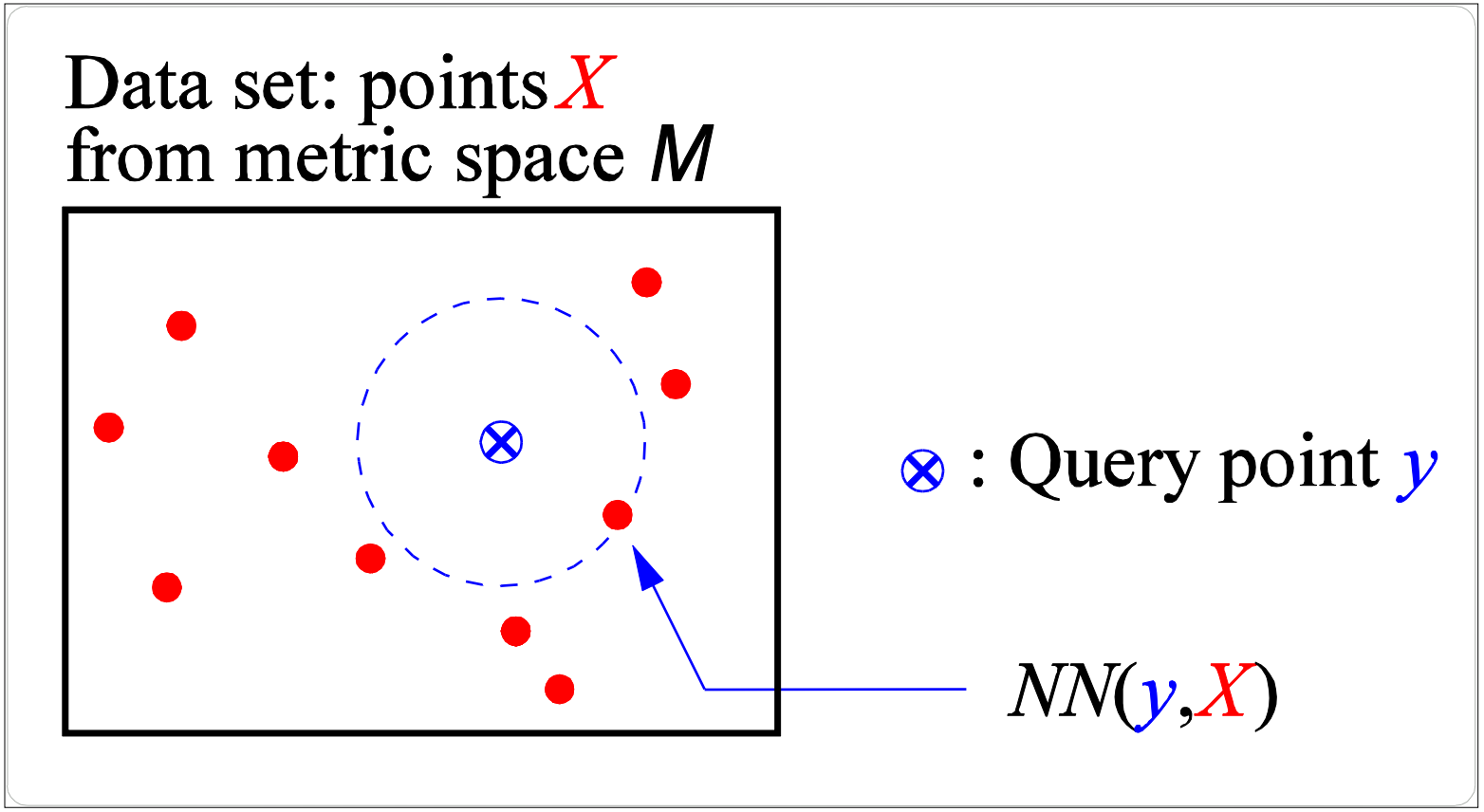


Starter Problem: Index

0 1 1 1 0 1 0 1 1 0 0 0 0 …	1258914

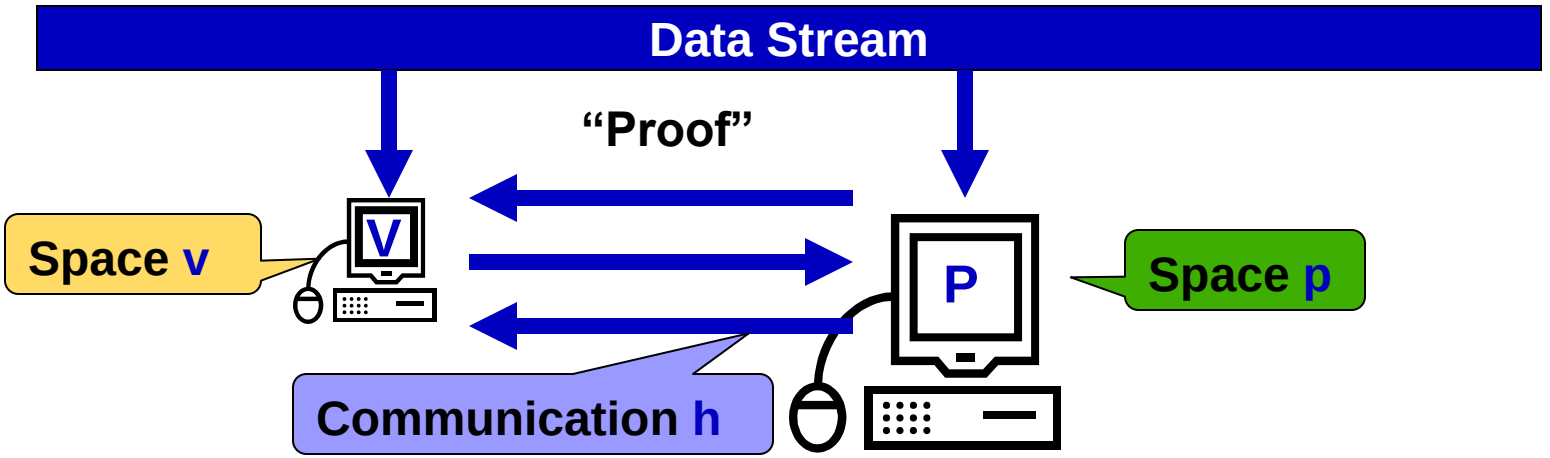
- Fundamental (hard) problem in data streams
  - Input is a length m binary string x followed by index y
  - Desired output is x[y]
  - Requires $\Omega(m)$ space even allowing error probability
- Can we find a protocol to allow recovery of arbitrary bits
  - Without having the verifier store the entire sequence?

Real problem: Nearest neighbor

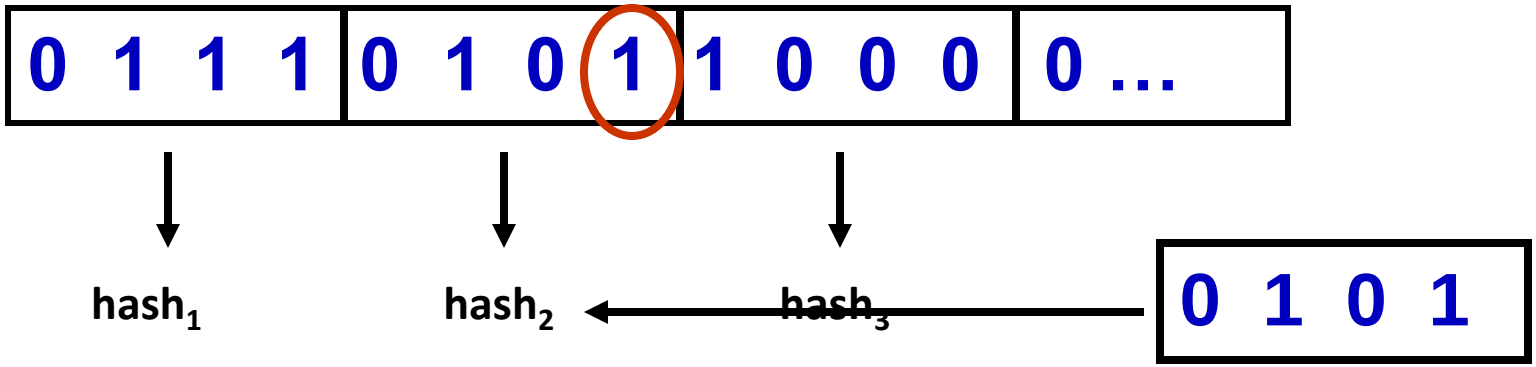


Parameters

- m data points (m very large)
  - Verifier V processes data using small space << m
  - Prover P processes data using space at least m
- V and P have a conversation to determine the answer
  - If P is honest, 0.99 probability that V accepts the answer
  - If P is dishonest, 0.99 probability that V rejects the answer
  - Measure the space used by V, P, communication used by both



Index: 1 Round Upper Bound

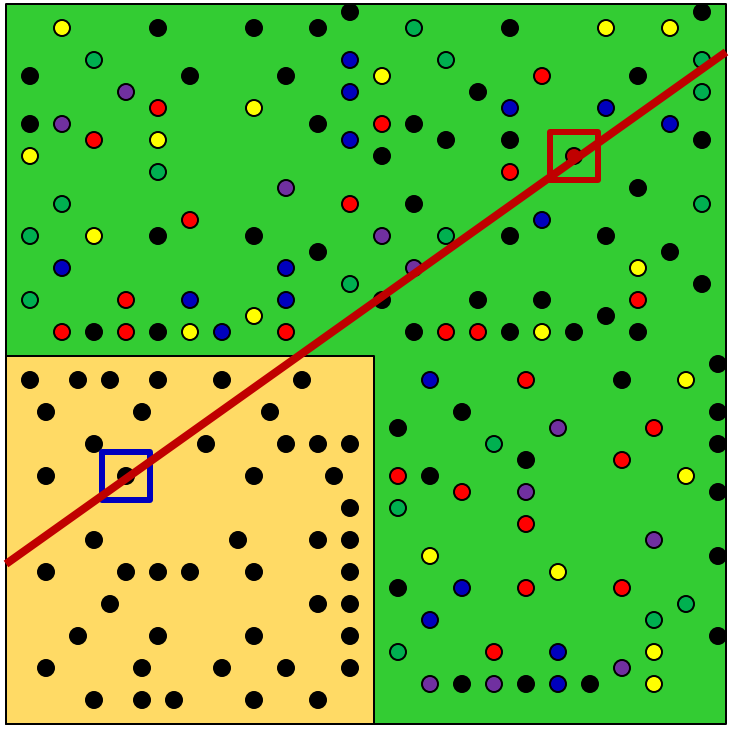


- Divide the bit string into blocks of H bits
- Verifier remembers a hash on each block
- After seeing index, Prover replays its block
- Verifier checks hash agrees, and outputs x[y]

- Cost: H bits of proof from the prover, V = m/H hashes
  - So HV = O(m log m), any point on tradeoff is possible

2 Round Index Protocol

Query point y



Data indexed in Boolean hypercube $\{0,1\}^b$

Extended to hypercube $\mathbf{F}^b$

Challenge line λ

Random point $r \in \mathbf{F}^b$

1. V picks r and evaluates low-degree extension of input at r to get q
2. V sends λ to P
3. P sends polynomial p' which is input restricted to λ
4. V checks that $p'(r) = q$, and outputs $p'(y)$

Streaming LDE Computation

- Given query point $r \in F^b$, evaluate extension of input at r
- Initialize: $z = 0$
- Update with impact of each data point $y=(y_1, \ldots y_b)$ in turn. Structure of polynomial means update causes

$$z \leftarrow z + \prod_{i=1}^{b} ((1-y_i)(1-r_i) + y_i r_i)$$

  - Lagrange polynomial, can be evaluated in small space

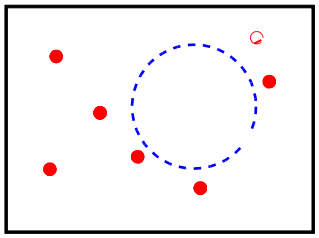
- Can be computed quickly, using appropriate precomputed look-up tables

Correctness and Cost

- Correctness of the protocol
  - If P is honest: V will always accept
  - If P is dishonest: V only accepts if $p'(r) = q$
    This happens with probability $b/|F|$: can make $|F|$ bigger
- Costs of the protocol
  - V's space: $O(b \log |F|) = O(\log n \log \log n)$ bits
  - P and V exchange λ and p' as $(b + 1)$ values in F,
    so communication cost is $O(\log n \log \log n)$ bits
  - Exponential improvement over one round
- Consequences: can do other computations via Index e.g. median
  - What about more complex functions?

Nearest Neighbour Search

- Basic idea: convert NNS into an (enormous) index problem
  - Work with input points in $[n]^d$
  - Assume all distances are multiples of $\varepsilon = 1/n^d$
- Let $B = \{\text{all distinct balls}\}$; note $|B| \leq n^{2d}$
  - Convert input points to **virtual set of balls** from B:
  - point $x \rightarrow$ all balls β such that $x \in \beta$
- V processes virtual stream σ through index protocol
- For query $y \in X$, P specifies point $z \in X$, claiming $z = NN(y,X)$
  - Show $ball(z,0) \in \sigma$ via Index Protocol
  - And $ball(z, dist(y, z)-\varepsilon) \notin \sigma$ via Index Protocol
- Protocol allows correct demonstration of nearest neighbour
- Drawback: blow-up of input size costs V a lot!



Practical Proof Protocol

- Exploit structure of the metric space containing the points
  - Let $\Phi(\beta,x)$ be the function that reports 1 iff x is in ball β
  - Goal: query the vector $v[\beta] = \sum_{x \text{ in input}} \Phi(\beta,x)$
  - $\Phi(\beta,x)$ has a simple circuit for common metrics (Hamming, L_1, L_2…)
  - "Arithmetize" the formula to compute distances
- Transform formula Φ to polynomial Φ' via

  $G_1 \wedge G_2 \mapsto G'_1 G'_2$ and $\quad G_1 \vee G_2 \mapsto 1-(1-G'_1)(1-G'_2)$
- Low-degree extension of v: $v'(B_1 \ldots B_{2d \log n}) = \sum_x \Phi'(B_1 \ldots B_{2d \log n}, x)$
  - Can then apply Index protocol to v' – v never materialized by P or V
- Final costs of the protocol:
  - Verifier can process each data point in time poly(d,log n)
  - Communication cost and verifier space both poly(d,log m,log n) bits

Concluding Remarks

- These protocols are truly practical
  - No, really, they are
- Also provide insight into the theory of Arthur-Merlin communication games

- Many open problems around this area
  - Extend to other data mining/machine learning problems
  - Prove lower bounds: some problems are hard
  - Evaluations on real data, optimization of implementations
  - Variant models: power of two provers…