Probabilistic Polynomials and Hamming Nearest Neighbors

Josh Alman   Ryan Williams

Stanford University

Workshop on Multi-dimensional Proximity Problems,
January 13, 2016

Hamming Nearest Neighbor Problem

Definition (Hamming Nearest Neighbor Problem)

Given a set D of n database points in $\{0,1\}^d$, we wish to preprocess D so that for queries $q \in \{0,1\}^d$, we answer a point $u \in D$ that differs from q in a minimum number of coordinates.

Hamming Nearest Neighbor Problem

Definition (Hamming Nearest Neighbor Problem)

Given a set D of n database points in $\{0,1\}^d$, we wish to preprocess D so that for queries $q \in \{0,1\}^d$, we answer a point $u \in D$ that differs from q in a minimum number of coordinates.

Curse of Dimensionality (Barkol, Rabani '00)

All solutions require either

- $2^{\Omega(d)}$ *size data structure (store all answers), or*
- $\Omega(n/\operatorname{polylog}(n))$ *query time (try all points).*

Hamming Nearest Neighbor Problem: Past Work

Past work has gotten around this problem in a variety of ways:

- *Approximate solutions:* find a point with distance within $(1+\epsilon)$ of the optimal
  - Lots of beautiful results and impact: hashing, dimensionality reduction, ...
  - "Curse of approximation": still requires $n^{\Omega(1/\epsilon^2)}$ space. [Andoni, Indyk, Patrascu '06]

Hamming Nearest Neighbor Problem: Past Work

Past work has gotten around this problem in a variety of ways:

- *Approximate solutions:* find a point with distance within $(1+\epsilon)$ of the optimal
  - Lots of beautiful results and impact: hashing, dimensionality reduction, ...
  - "Curse of approximation": still requires $n^{\Omega(1/\epsilon^2)}$ space. [Andoni, Indyk, Patrascu '06]
- 'Planted' case: All vectors are random except one pair with distance much smaller than expected; find the planted pair among the n vectors
  - $O(n^{1.62})$ time algorithm, independent of dimension. [G. Valiant '12]

Batch Hamming Nearest Neighbor Problem

Definition (Batch Hamming Nearest Neighbor Problem)

Given a set D of n database points in $\{0,1\}^d$, and a set Q of n query points in $\{0,1\}^d$, find the HNN in D for each point in Q.

Batch Hamming Nearest Neighbor Problem

Definition (Batch Hamming Nearest Neighbor Problem)

Given a set D of n database points in $\{0,1\}^d$, and a set Q of n query points in $\{0,1\}^d$, find the HNN in D for each point in Q.

Lower bounds no longer apply, but still best previously known solutions take either:

- $n \cdot 2^{\Omega(d)}$ time (build a table of all answers), or
- $n^2 \cdot d^{\Omega(1)}$ time (try all pairs).

Batch Hamming Nearest Neighbor Problem: Our Result

Theorem (AW '15)

Let $D \subseteq \{0,1\}^d$ be a database of n vectors of dimension $d = c \log n$, where c can be a function of n. Any batch of n Hamming nearest neighbor queries on D can be answered in randomized $n^{2-1/O(c \log^2 c)}$ time, whp.

Batch Hamming Nearest Neighbor Problem: Our Result

Theorem (AW '15)

Let $D \subseteq \{0,1\}^d$ be a database of n vectors of dimension $d = c \log n$, where c can be a function of n. Any batch of n Hamming nearest neighbor queries on D can be answered in randomized $n^{2-1/O(c \log^2 c)}$ time, whp.

- If $d = O(\log n)$, then the algorithm runs in *truly subquadratic* time: $n^{2-\epsilon}$, for some $\epsilon > 0$.

Batch Hamming Nearest Neighbor Problem: Our Result

Theorem (AW '15)

Let $D \subseteq \{0,1\}^d$ be a database of n vectors of dimension $d = c \log n$, where c can be a function of n. Any batch of n Hamming nearest neighbor queries on D can be answered in randomized $n^{2-1/O(c \log^2 c)}$ time, whp.

- If $d = O(\log n)$, then the algorithm runs in *truly subquadratic* time: $n^{2-\epsilon}$, for some $\epsilon > 0$.
- Improves on the trivial algorithm when $d = o(\log^2(n)/\log\log^2(n))$.

Batch Hamming Nearest Neighbor Problem: Our Result

Theorem (AW '15)

Let $D \subseteq \{0,1\}^d$ be a database of n vectors of dimension $d = c \log n$, where c can be a function of n. Any batch of n Hamming nearest neighbor queries on D can be answered in randomized $n^{2-1/O(c \log^2 c)}$ time, whp.

- If $d = O(\log n)$, then the algorithm runs in *truly subquadratic* time: $n^{2-\epsilon}$, for some $\epsilon > 0$.
- Improves on the trivial algorithm when $d = o(\log^2(n)/\log\log^2(n))$.
- Algorithm technique: Compute Hamming distances using Efficiently Computable Low-Degree Probabilistic Polynomials (Very different techniques from past work)

Batch Hamming Nearest Neighbor Problem: Our Result

Theorem (AW '15)

Let $D \subseteq \{0,1\}^d$ be a database of n vectors of dimension $d = c \log n$, where c can be a function of n. Any batch of n Hamming nearest neighbor queries on D can be answered in randomized $n^{2-1/O(c \log^2 c)}$ time, whp.

- If $d = O(\log n)$, then the algorithm runs in *truly subquadratic* time: $n^{2-\epsilon}$, for some $\epsilon > 0$.

Theorem (AW '15)

Suppose there is $\epsilon > 0$ such that for all constant c, Batch HNN can be solved in $2^{o(d)} \cdot n^{2-\epsilon}$ time on a set of n points in $\{0,1\}^{c \log n}$. Then the Strong Exponential Time Hypothesis is false.

Polynomials that Compute Boolean Functions

Let R be a ring (can be $\mathbb{Z}_m, \mathbb{Z}, \mathbb{Q}, \mathbb{R}, \ldots$). A polynomial p in n variables over R computes the boolean function $f : \{0,1\}^n \to \{0,1\}$ if for each $x \in \{0,1\}^n$ we have $p(x) = f(x)$.

Polynomials that Compute Boolean Functions

Let R be a ring (can be $\mathbb{Z}_m, \mathbb{Z}, \mathbb{Q}, \mathbb{R}, \ldots$). A polynomial p in n variables over R computes the boolean function $f : \{0,1\}^n \to \{0,1\}$ if for each $x \in \{0,1\}^n$ we have $p(x) = f(x)$.

Example (OR function)

$$OR(x_1, x_2, \ldots, x_n) = \begin{cases} 0 & \text{if } x_1 = x_2 = \cdots = x_n = 0 \\ 1 & \text{otherwise.} \end{cases}$$

Then,

$$OR(x) = p(x) := 1 - (1 - x_1)(1 - x_2) \cdots (1 - x_n)$$

Polynomials that Compute Boolean Functions

Let R be a ring (can be $\mathbb{Z}_m, \mathbb{Z}, \mathbb{Q}, \mathbb{R}, \ldots$). A polynomial p in n variables over R computes the boolean function $f : \{0,1\}^n \to \{0,1\}$ if for each $x \in \{0,1\}^n$ we have $p(x) = f(x)$.

Example (OR function)

$$OR(x_1, x_2, \ldots, x_n) = \begin{cases} 0 & \text{if } x_1 = x_2 = \cdots = x_n = 0 \\ 1 & \text{otherwise.} \end{cases}$$

Then,

$$OR(x) = p(x) := 1 - (1 - x_1)(1 - x_2) \cdots (1 - x_n)$$

The polynomial p has degree n, and 2^n terms when expanded out.

Polynomials that Compute Boolean Functions

Let R be a ring (can be $\mathbb{Z}_m, \mathbb{Z}, \mathbb{Q}, \mathbb{R}, \ldots$). A polynomial p in n variables over R computes the boolean function $f : \{0,1\}^n \to \{0,1\}$ if for each $x \in \{0,1\}^n$ we have $p(x) = f(x)$.

Example (OR function)

$$OR(x_1, x_2, \ldots, x_n) = \begin{cases} 0 & \text{if } x_1 = x_2 = \cdots = x_n = 0 \\ 1 & \text{otherwise.} \end{cases}$$

Then,

$$OR(x) = p(x) := 1 - (1 - x_1)(1 - x_2)\cdots(1 - x_n)$$

The polynomial p has degree n, and 2^n terms when expanded out.

Note: We never need to take powers of a variable greater than 1, since $x_i = x_i^2$ when $x_i \in \{0,1\}$. (We only need to look at multilinear polynomials)

Probabilistic Polynomial

Let $f : \{0,1\}^n \to \{0,1\}$ be any Boolean function on n variables.

Definition (Probabilistic Polynomial)

A *probabilistic polynomial over* R *for* f *with error* ϵ *and degree* d is a distribution $\mathcal{D}$ of degree-d polynomials over R with the property that for each $x \in \{0,1\}^n$,

$$\Pr_{p \sim \mathcal{D}}[p(x) = f(x)] \geq 1 - \epsilon.$$

Note: The probability is only over the polynomial p, not over the input x.

Probabilistic Polynomial Example: OR over $R = \mathbb{Z}$

[Aspnes, Beigel, Furst, Rudich '93]

Set $S_0 = \{1, 2, \ldots, n\}$ and construct subsets

$$S_0 \supseteq S_1 \supseteq S_2 \cdots \supseteq S_{\log_2(n)+1}$$

such that each element of S_i is included in S_{i+1} with probability $1/2$.

Let $p_i(x) = \sum_{j \in S_i} x_j$.

Our probabilistic polynomial for OR is

$$p(x) = 1 - \prod_i (1 - p_i(x))$$

Probabilistic Polynomial Example: OR over $R = \mathbb{Z}$

[Aspnes, Beigel, Furst, Rudich '93]

Set $S_0 = \{1, 2, \ldots, n\}$ and construct subsets

$$S_0 \supseteq S_1 \supseteq S_2 \cdots \supseteq S_{\log_2(n)+1}$$

such that each element of S_i is included in S_{i+1} with probability $1/2$.

Let $p_i(x) = \sum_{j \in S_i} x_j$.

Our probabilistic polynomial for OR is

$$p(x) = 1 - \prod_i (1 - p_i(x))$$

- If $x = (0, \ldots, 0)$, then $p_j(x) \equiv 0$ and $p(x) = 0$.
- If $x \neq (0, \ldots, 0)$ then we want there to be a j such that $p_j(x) = 1$ with some (constant) probability.

Probabilistic Polynomial Example: OR over $R = \mathbb{Z}$

[Aspnes, Beigel, Furst, Rudich '93]

Set $S_0 = \{1, 2, \dots, n\}$ and construct subsets

$$S_0 \supseteq S_1 \supseteq S_2 \cdots \supseteq S_{\log_2(n)+1}$$

such that each element of S_i is included in S_{i+1} with probability $1/2$.

Let $p_i(x) = \sum_{j \in S_i} x_j$.

Our probabilistic polynomial for OR is

$$p(x) = 1 - \prod_i (1 - p_i(x))$$

- If $x = (0, \dots, 0)$, then $p_j(x) \equiv 0$ and $p(x) = 0$.
- If $x \neq (0, \dots, 0)$ then we want there to be a j such that $p_j(x) = 1$ with some (constant) probability.

Probabilistic Polynomial Example: OR over $R = \mathbb{Z}$

[Aspnes, Beigel, Furst, Rudich '93]

Set $S_0 = \{1, 2, \ldots, n\}$ and construct subsets

$$S_0 \supseteq S_1 \supseteq S_2 \cdots \supseteq S_{\log_2(n)+1}$$

such that each element of S_i is included in S_{i+1} with probability $1/2$.

Let $p_i(x) = \sum_{j \in S_i} x_j$.

Our probabilistic polynomial for OR is

$$p(x) = 1 - \prod_i (1 - p_i(x))$$

- If $x = (0, \ldots, 0)$, then $p_j(x) \equiv 0$ and $p(x) = 0$.
- If $x \neq (0, \ldots, 0)$ then we want there to be a j such that $p_j(x) = 1$ with some (constant) probability.

Probabilistic Polynomial Example: OR over $R = \mathbb{Z}$

[Aspnes, Beigel, Furst, Rudich '93]

Set $S_0 = \{1, 2, \ldots, n\}$ and construct subsets

$$S_0 \supseteq S_1 \supseteq S_2 \cdots \supseteq S_{\log_2(n)+1}$$

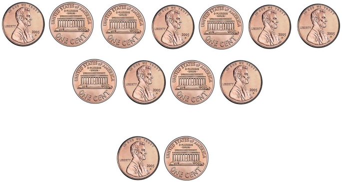
such that each element of S_i is included in S_{i+1} with probability $1/2$.

Let $p_i(x) = \sum_{j \in S_i} x_j$.

Our probabilistic polynomial for OR is

$$p(x) = 1 - \prod_i (1 - p_i(x))$$

- If $x = (0, \ldots, 0)$, then $p_j(x) \equiv 0$ and $p(x) = 0$.
- If $x \neq (0, \ldots, 0)$ then we want there to be a j such that $p_j(x) = 1$ with some (constant) probability.



Probabilistic Polynomial Example: OR over $R = \mathbb{Z}$

[Aspnes, Beigel, Furst, Rudich '93]

Set $S_0 = \{1, 2, \ldots, n\}$ and construct subsets

$$S_0 \supseteq S_1 \supseteq S_2 \cdots \supseteq S_{\log_2(n)+1}$$

such that each element of S_i is included in S_{i+1} with probability $1/2$.

Let $p_i(x) = \sum_{j \in S_i} x_j$.

Our probabilistic polynomial for OR is

$$p(x) = 1 - \prod_i (1 - p_i(x))$$

- If $x = (0, \ldots, 0)$, then $p_j(x) \equiv 0$ and $p(x) = 0$.
- If $x \neq (0, \ldots, 0)$ then we want there to be a j such that $p_j(x) = 1$ with some (constant) probability.

Probabilistic Polynomial Example: OR over $R = \mathbb{Z}$

[Aspnes, Beigel, Furst, Rudich '93]

Set $S_0 = \{1, 2, \ldots, n\}$ and construct subsets

$$S_0 \supseteq S_1 \supseteq S_2 \cdots \supseteq S_{\log_2(n)+1}$$

such that each element of S_i is included in S_{i+1} with probability $1/2$.

Let $p_i(x) = \sum_{j \in S_i} x_j$.

Our probabilistic polynomial for OR is

$$p(x) = 1 - \prod_i (1 - p_i(x))$$

- $O(\log(n))$ degree polynomial for OR with $\epsilon = 2/3$.
- Can augment to degree $O(\log(n)\log(1/\epsilon))$ for any $\epsilon > 0$ (use the fact that the error is one-sided).

Probabilistic Polynomials for MAJORITY

Notation: For $x \in \{0,1\}^n$, write $|x| = \sum_{i=1}^n x_i$.

$MAJORITY(x) = 1$ iff $|x| \geq n/2$.

Probabilistic Polynomials for MAJORITY

Notation: For $x \in \{0,1\}^n$, write $|x| = \sum_{i=1}^{n} x_i$.

$MAJORITY(x) = 1$ iff $|x| \geq n/2$.

Theorem (Razborov, Smolensky '87)

A probabilistic polynomial with ϵ error for MAJORITY requires degree $\Omega(\sqrt{n \log(1/\epsilon)})$.

Probabilistic Polynomials for MAJORITY

Notation: For $x \in \{0,1\}^n$, write $|x| = \sum_{i=1}^n x_i$.

$MAJORITY(x) = 1$ iff $|x| \geq n/2$.

Theorem (Razborov, Smolensky '87)

A probabilistic polynomial with ϵ error for MAJORITY requires degree $\Omega(\sqrt{n \log(1/\epsilon)})$.

Theorem (AW '15)

There is a probabilistic polynomial for MAJORITY on n variables with error ϵ and degree $O(\sqrt{n \log(1/\epsilon)})$.

Probabilistic Polynomials for MAJORITY

Notation: For $x \in \{0,1\}^n$, write $|x| = \sum_{i=1}^n x_i$.

$MAJORITY(x) = 1$ iff $|x| \geq n/2$.

Theorem (Razborov, Smolensky '87)

A probabilistic polynomial with ϵ error for MAJORITY requires degree $\Omega(\sqrt{n \log(1/\epsilon)})$.

Theorem (AW '15)

There is a probabilistic polynomial for MAJORITY on n variables with error ϵ and degree $O(\sqrt{n \log(1/\epsilon)})$.

We will actually look at the threshold function:
$TH_\theta(x) = 1$ iff $|x|/n \geq \theta$. In particular, $MAJORITY = TH_{1/2}$.

THRESHOLD: Recursive Intuition

Two cases depending on how close $|x|/n$ is to θ (whether or not it is within $\delta = \Theta(\sqrt{\log(1/\epsilon)/n})$):

THRESHOLD: Recursive Intuition

Two cases depending on how close $|x|/n$ is to θ (whether or not it is within $\delta = \Theta(\sqrt{\log(1/\epsilon)/n})$):

- If $|x|/n \notin [\theta - \delta, \theta + \delta]$, then if we construct a new smaller vector $\tilde{x}$ by sampling $1/10$ of the entries of x, it is likely that $|\tilde{x}|/(n/10)$ lies on the same side of θ as $|x|/n$ (by Chernoff-Hoeffding).

THRESHOLD: Recursive Intuition

Two cases depending on how close $|x|/n$ is to θ (whether or not it is within $\delta = \Theta(\sqrt{\log(1/\epsilon)/n})$):

- If $|x|/n \notin [\theta - \delta, \theta + \delta]$, then if we construct a new smaller vector $\tilde{x}$ by sampling $1/10$ of the entries of x, it is likely that $|\tilde{x}|/(n/10)$ lies on the same side of θ as $|x|/n$ (by Chernoff-Hoeffding).
- If $|x|/n \in [\theta - \delta, \theta + \delta]$, we can use an exact polynomial of degree $O(n\delta) = O(\sqrt{n \log(1/\epsilon)})$ (by polynomial interpolation) that is guaranteed to give the correct answer.

THRESHOLD: Recursive Intuition

Two cases depending on how close $|x|/n$ is to θ (whether or not it is within $\delta = \Theta(\sqrt{\log(1/\epsilon)/n})$):

- If $|x|/n \notin [\theta - \delta, \theta + \delta]$, then if we construct a new smaller vector $\tilde{x}$ by sampling $1/10$ of the entries of x, it is likely that $|\tilde{x}|/(n/10)$ lies on the same side of θ as $|x|/n$ (by Chernoff-Hoeffding).
- If $|x|/n \in [\theta - \delta, \theta + \delta]$, we can use an exact polynomial of degree $O(n\delta) = O(\sqrt{n \log(1/\epsilon)})$ (by polynomial interpolation) that is guaranteed to give the correct answer.
- To decide which of the two cases we are in, we can use $TH_{\theta+\delta}(\tilde{x})$ and $TH_{\theta-\delta}(\tilde{x})$.

From Probabilistic Polynomial to Hamming Distance Algorithm

Given an efficient (small number of monomials) polynomial, we can evaluate it on many points quickly:

Lemma (R. Williams '14)

Given a polynomial $P(x_1, \ldots, x_d, y_1, \ldots, y_d)$ with at most $n^{0.17}$ monomials, and two sets of n inputs $A = \{a_1, \ldots, a_n\} \subseteq \{0,1\}^d$, $B = \{b_1, \ldots, b_n\} \subseteq \{0,1\}^d$, we can evaluate P on all pairs $(a_i, b_j) \in A \times B$ in $\tilde{O}(n^2)$ time.

From Probabilistic Polynomial to Hamming Distance Algorithm

Given an efficient (small number of monomials) polynomial, we can evaluate it on many points quickly:

Lemma (R. Williams '14)

Given a polynomial $P(x_1, \ldots, x_d, y_1, \ldots, y_d)$ with at most $n^{0.17}$ monomials, and two sets of n inputs $A = \{a_1, \ldots, a_n\} \subseteq \{0,1\}^d$, $B = \{b_1, \ldots, b_n\} \subseteq \{0,1\}^d$, we can evaluate P on all pairs $(a_i, b_j) \in A \times B$ in $\tilde{O}(n^2)$ time.

- Beats the trivial runtime of $\Omega(n^{2.17})$ time.
- Since we want a subquadratic algorithm, we can't just let A, B be our sets of vectors.
- Instead, group our vectors into n/s groups of size s. Each element of A or B will correspond to a group.

Hamming distance subproblem

We will use this to solve the following sub-problem of Batch HNN:

Definition (Hamming distance problem)

Given an integer k and two collections of s vectors of dimension d as input, output 1 iff there is a pair of vectors (one from each collection) with Hamming distance at most k.

Polynomial for Hamming distance problem over $\mathbb{F}_2$

- Let $x_1, \ldots, x_s$ and $y_1, \ldots, y_s$ be the two collections of vectors. We will write $x_{i,j}$ for the jth variable of vector x_i.

Polynomial for Hamming distance problem over $\mathbb{F}_2$

- Let $x_1, \ldots, x_s$ and $y_1, \ldots, y_s$ be the two collections of vectors. We will write $x_{i,j}$ for the jth variable of vector x_i.
- Let $\mathcal{D}_d$ be the probabilistic polynomial of degree $O(\sqrt{d \log(1/\epsilon)})$ for the threshold function $TH_{(k+1)/d}$ on d inputs. Sample $p \sim \mathcal{D}_d$ with error $\epsilon = 1/s^3$.

Polynomial for Hamming distance problem over $\mathbb{F}_2$

- Let $x_1, \ldots, x_s$ and $y_1, \ldots, y_s$ be the two collections of vectors. We will write $x_{i,j}$ for the jth variable of vector x_i.
- Let $\mathcal{D}_d$ be the probabilistic polynomial of degree $O(\sqrt{d \log(1/\epsilon)})$ for the threshold function $TH_{(k+1)/d}$ on d inputs. Sample $p \sim \mathcal{D}_d$ with error $\epsilon = 1/s^3$.
- Choose a uniform random subset $R \subseteq \{1, 2, \ldots, s\}^2$

Polynomial for Hamming distance problem over $\mathbb{F}_2$

- Let $x_1, \ldots, x_s$ and $y_1, \ldots, y_s$ be the two collections of vectors. We will write $x_{i,j}$ for the jth variable of vector x_i.
- Let $\mathcal{D}_d$ be the probabilistic polynomial of degree $O(\sqrt{d \log(1/\epsilon)})$ for the threshold function $TH_{(k+1)/d}$ on d inputs. Sample $p \sim \mathcal{D}_d$ with error $\epsilon = 1/s^3$.
- Choose a uniform random subset $R \subseteq \{1, 2, \ldots, s\}^2$

Our polynomial is:

$$q(x_1, y_1, \ldots, x_s, y_s) := \sum_{(i,j)\in R} (1 + p(x_{i,1} + y_{j,1}, \ldots, x_{i,d} + y_{j,d})).$$

Polynomial for Hamming distance problem over $\mathbb{F}_2$: Correctness

$$q(x_1, y_1, \ldots, x_s, y_s) := \sum_{(i,j)\in R} (1 + p(x_{i,1} + y_{j,1}, \ldots, x_{i,d} + y_{j,d})).$$

- Since we are working over $\mathbb{F}_2$, the number of 1s in the vector $(x_{i,1} + y_{j,1}, \ldots, x_{i,d} + y_{j,d})$ is the Hamming distance between x_i and y_j.

Polynomial for Hamming distance problem over $\mathbb{F}_2$: Correctness

$$q(x_1, y_1, \ldots, x_s, y_s) := \sum_{(i,j)\in R} (1 + p(x_{i,1} + y_{j,1}, \ldots, x_{i,d} + y_{j,d})).$$

- Since we are working over $\mathbb{F}_2$, the number of 1s in the vector $(x_{i,1} + y_{j,1}, \ldots, x_{i,d} + y_{j,d})$ is the Hamming distance between x_i and y_j.
- $1 + p(x_{i,1} + y_{j,1}, \ldots, x_{i,d} + y_{j,d})$ is 1 iff x_i and y_j have Hamming distance at most k (assuming p is correct)

Polynomial for Hamming distance problem over $\mathbb{F}_2$: Correctness

$$q(x_1, y_1, \ldots, x_s, y_s) := \sum_{(i,j)\in R} (1 + p(x_{i,1} + y_{j,1}, \ldots, x_{i,d} + y_{j,d})).$$

- Since we are working over $\mathbb{F}_2$, the number of 1s in the vector $(x_{i,1} + y_{j,1}, \ldots, x_{i,d} + y_{j,d})$ is the Hamming distance between x_i and y_j.
- $1 + p(x_{i,1} + y_{j,1}, \ldots, x_{i,d} + y_{j,d})$ is 1 iff x_i and y_j have Hamming distance at most k (assuming p is correct)
- If all the x_i and y_j have Hamming distance $> k$, then the sum is 0. Otherwise, it is 0 or 1 with $1/2$ chance each (based on our choice of R)

Polynomial for Hamming distance problem over $\mathbb{F}_2$: Correctness

$$q(x_1, y_1, \ldots, x_s, y_s) := \sum_{(i,j)\in R} (1 + p(x_{i,1} + y_{j,1}, \ldots, x_{i,d} + y_{j,d})).$$

- Since we are working over $\mathbb{F}_2$, the number of 1s in the vector $(x_{i,1} + y_{j,1}, \ldots, x_{i,d} + y_{j,d})$ is the Hamming distance between x_i and y_j.
- $1 + p(x_{i,1} + y_{j,1}, \ldots, x_{i,d} + y_{j,d})$ is 1 iff x_i and y_j have Hamming distance at most k (assuming p is correct)
- If all the x_i and y_j have Hamming distance $> k$, then the sum is 0. Otherwise, it is 0 or 1 with $1/2$ chance each (based on our choice of R)
- Since the error is one-sided, we can amplify to get as high a success probability as we want.

Solving Batch Hamming Nearest Neighbor

Two more steps:

- Hamming distance problem (is there a pair with distance $\leq k$) polynomial $\Rightarrow$ algorithm.
- Hamming distance problem algorithm $\Rightarrow$ Batch Hamming nearest neighbor (for each vector, find its nearest neighbor) algorithm.

Hamming distance problem polynomial $\Rightarrow$ algorithm

Lemma

Given a threshold k, and subsets $R, B \subseteq \{0,1\}^d$ with $|R| = |B| = n$ and $d = c \log n$, we can find a $v \in R$ and $u \in B$ whose Hamming distance is $\leq k$ in time $n^{2-1/O(c \log^2 c)}$ (or determine that none exist).

Hamming distance problem polynomial $\Rightarrow$ algorithm

Lemma

Given a threshold k, and subsets $R, B \subseteq \{0,1\}^d$ with $|R| = |B| = n$ and $d = c \log n$, we can find a $v \in R$ and $u \in B$ whose Hamming distance is $\leq k$ in time $n^{2-1/O(c \log^2 c)}$ (or determine that none exist).

- Partition each of R and B into n/s groups of size $s = n^{1/O(c \log^2 c)}$.

Hamming distance problem polynomial $\Rightarrow$ algorithm

Lemma

Given a threshold k, and subsets $R, B \subseteq \{0,1\}^d$ with $|R| = |B| = n$ and $d = c \log n$, we can find a $v \in R$ and $u \in B$ whose Hamming distance is $\leq k$ in time $n^{2-1/O(c \log^2 c)}$ (or determine that none exist).

- Partition each of R and B into n/s groups of size $s = n^{1/O(c \log^2 c)}$.
- Our probabilistic polynomial on each group has degree $O(\sqrt{d \log s})$. Hence the number of monomials is $\binom{2d}{\sqrt{d \log s}}$, which is $\leq n^{0.17}$ for a suitable choice of constants.

Hamming distance problem polynomial $\Rightarrow$ algorithm

Lemma

Given a threshold k, and subsets $R, B \subseteq \{0,1\}^d$ with $|R| = |B| = n$ and $d = c \log n$, we can find a $v \in R$ and $u \in B$ whose Hamming distance is $\leq k$ in time $n^{2-1/O(c \log^2 c)}$ (or determine that none exist).

- Partition each of R and B into n/s groups of size $s = n^{1/O(c \log^2 c)}$.
- Our probabilistic polynomial on each group has degree $O(\sqrt{d \log s})$. Hence the number of monomials is $\binom{2d}{\sqrt{d \log s}}$, which is $\leq n^{0.17}$ for a suitable choice of constants.
- Using the fast evaluation lemma, we can evaluate on all pairs of groups in $n^{2-1/O(c \log^2 c)}$ time.

Hamming distance problem polynomial $\Rightarrow$ algorithm

Lemma

Given a threshold k, and subsets $R, B \subseteq \{0,1\}^d$ with $|R| = |B| = n$ and $d = c \log n$, we can find a $v \in R$ and $u \in B$ whose Hamming distance is $\leq k$ in time $n^{2-1/O(c \log^2 c)}$ (or determine that none exist).

- Partition each of R and B into n/s groups of size $s = n^{1/O(c \log^2 c)}$.
- Our probabilistic polynomial on each group has degree $O(\sqrt{d \log s})$. Hence the number of monomials is $\binom{2d}{\sqrt{d \log s}}$, which is $\leq n^{0.17}$ for a suitable choice of constants.
- Using the fast evaluation lemma, we can evaluate on all pairs of groups in $n^{2-1/O(c \log^2 c)}$ time.
- Brute force within a pair of groups which has a close pair to find the vectors.

Hamming distance problem algorithm $\Rightarrow$ Batch Hamming nearest neighbor

Lemma

If the Hamming distance problem can be solved in $T(n, d)$ time, then the Batch Hamming nearest neighbor problem can be solved in $O(ndT(\sqrt{n}, d))$ time.

Hamming distance problem algorithm $\Rightarrow$ Batch Hamming nearest neighbor

Lemma

If the Hamming distance problem can be solved in $T(n, d)$ time, then the Batch Hamming nearest neighbor problem can be solved in $O(ndT(\sqrt{n}, d))$ time.

- Partition each of D and Q into n/s groups of size $s = \sqrt{n}$.
- For k from $0, 1, 2, \ldots, d - 1$:
- Call the Hamming distance problem algorithm on each pair of a group from D and a group from Q. If a pair $(u, v) \in Q \times D$ is found, then v is a nearest neighbor for u. Remove u from Q and continue.
- There are at most n calls that do not return a vector pair for each k, so dn total such calls.
- There are at most n calls that return a vector pair since we remove each vector from Q once we find a pair for it.

Putting it all together

Combining our lemmas yields:

Theorem (AW '15)

Let $D \subseteq \{0,1\}^d$ *be a database of* n *vectors of dimension* $d = c \log n$*, where* c *can be a function of* n*. Any batch of* n *Hamming nearest neighbor queries on* D *can be answered in randomized* $n^{2-1/O(c \log^2 c)}$ *time, whp.*

Future Work?

Future Work?

- Better dependence on c in the exponent
  - A similar 'Polynomial Method' algorithm for Orthogonal Vectors has runtime $n^{2-1/O(\log c)}$.

Future Work?

- Better dependence on c in the exponent
  - A similar 'Polynomial Method' algorithm for Orthogonal Vectors has runtime $n^{2-1/O(\log c)}$.
- Use the 'Polynomial Method' for other problems
  - Has already been used for APSP, OV, SAT, CSPs, string matching...

Future Work?

- Better dependence on c in the exponent
  - A similar 'Polynomial Method' algorithm for Orthogonal Vectors has runtime $n^{2-1/O(\log c)}$.
- Use the 'Polynomial Method' for other problems
  - Has already been used for APSP, OV, SAT, CSPs, string matching...
- Derandomize the MAJORITY probabilistic polynomial
  - Timothy Chan and Ryan Williams derandomized most 'Polynomial Method' algorithms at SODA'16.
  - Would also give circuit lower bounds (e.g. for $THR \circ THR$ circuits)

Future Work?

- Better dependence on c in the exponent
  - A similar 'Polynomial Method' algorithm for Orthogonal Vectors has runtime $n^{2-1/O(\log c)}$.
- Use the 'Polynomial Method' for other problems
  - Has already been used for APSP, OV, SAT, CSPs, string matching...
- Derandomize the MAJORITY probabilistic polynomial
  - Timothy Chan and Ryan Williams derandomized most 'Polynomial Method' algorithms at SODA'16.
  - Would also give circuit lower bounds (e.g. for $THR \circ THR$ circuits)
- Other ways to quickly evaluate polynomials
  - Feels strange to use matrix multiplication instead of FFT
  - That said, fast MM used here is not necessarily impractical!