Algorithmic Coding theory presentation: Noisy interpolating sets for low degree polynomials Dvir Shpilka [DS08]

Bijayan Ray

November 27, 2023

Contents

The problem I

Definition 1.1

$S = (a_1, \cdots a_m) \in (\mathbb{F}^n)^m$ be a list of points in $\mathbb{F}^n$. We say that S is an (n, d, ϵ)-Noisy interpolating set (NIS) if there exists an algorithm A_S such that for every $q \in F_d[x_1, \cdots x_n]$ and for every vector $e = (e_1, \cdots e_m) \in \mathbb{F}^m$ such that $|\{i \in [m] : e_i \neq 0\}| \leq \epsilon \cdot m$, the algorithm A_S, when given an input the list of values $(q(a_1) + e_1, \cdots, q(a_m) + e_m)$, outputs the polynomial q (as a list of coefficients).

We say that S is a proper NIS if the points $a_1, \cdots a_m$ are distinct. If S is a proper NIS we can treat it as a subset $S \subset \mathbb{F}^n$.

The problem II

Definition 1.2

$S = (S^{(n)})_{n \in \mathbb{N}}$ be a sequence such that for every $n \in \mathbb{N}$ we have that $S^{(n)}$ is an (n, d, ϵ)-NIS. We say that S has an efficient interpolation algorithm if there exists a polynomial time algorithm $M(n, L)$ that takes an input an integer n and a list L of values in $\mathbb{F}$ such that the restriction $M(n, \cdot)$ has the same behaviour as the algorithm $A_{S^{(n)}}$ described above.

Problem 1.1

The problem is to compute the NIS of a set of d-degree polynomials efficiently.

Main theorems I

Definition 2.1

1. *Set addition*
$$A + B = \{a + b | a \in A, b \in B\}$$

2. *$A \boxplus B$ is the list defined as:*
$$A \boxplus B = (a_i + b_j)_{i \in [m], j \in [l]} \in (\mathbb{F}^n)^{ml}$$

Theorem 2.1 (NIS)

Consider $0 < \epsilon_1 \leq 1/2$ be a real number and S_1 be an $(n, 1, \epsilon_1)$-NIS and for each $d > 1$ take $S_d = S_{d-1} \boxplus S_1$. Then for every $d > 1$ the set S_d is an (n, d, ϵ_d)-NIS with $\epsilon_d = (\epsilon/2)^d$. Moreover, if S_1 has an efficient interpolation algorithm, then so does S_d.

Main theorems II

Corollary 2.1 (NIS)

For every prime field $\mathbb{F}$ and for every $d > 0$ there exists an $\epsilon > 0$ and a collection $\mathcal{S} = (S^n)_{n\in\mathbb{N}}$ such that for every $n \in \mathbb{N}$, $S^{(n)}$ is an (n, d, ϵ)-NIS and such that $\mathcal{S}$ has an efficient interpolation algorithm. Moreover, for each $n \in \mathbb{N}$ we have $|S^{(n)}| = O(n^d)$ and it is possible to generate the set $S^{(n)}$ in time poly(n).

Main theorems III

Definition 2.2 (Condition $\star_k$)

Consider $S \in \mathbb{F}^n$, $S = \{0\}$ and for each $d \geq 1$ consider $S_d = S_{d-1} + S$. Consider $k > 0$ be an integer. For each $x \in S_d$ consider

$$N_d(x) = |\{b \in S | s \in S_{d-1} + b\}|$$

We say that S satisfies condition $\star_k$ if for every $0 < d \leq k$ we have

$$|\{x \in S_d | N_d(x) > d\}| \leq |S_{d-2}|$$

condition satisfied.

Main theorems IV

Theorem 2.2 (Proper-NIS)

Consider $0 < \epsilon_1 \leq 1/2$ *be a real number and* $k > 0$ *be an integer. There exists a constant* C_0*, depending only on* ϵ *and* k*, such that for all* $n > C_0$ *the following holds: For every proper* $(n, 1, \epsilon_1)$*-NIS set* S_1 *and for each* $d > 1$ *denote* $S_d = S_{d-1} + S_1$*. Suppose* S_1 *satisfies the condition* $\star_s$ *(definition 2.2). Then for every* $1 < d \leq k$ *the set* S_d *is a (proper)* (n, d, ϵ_d)*-NIS with*

$$\epsilon_d = \frac{1}{d!} \cdot \left(\frac{\epsilon_1}{3}\right)^d$$

Moreover, if S_1 *has an efficient interpolation algorithm, then so does* S_d*.*

Proof.

We don't include the proof of this theorem in this presentation slides, however one can refer the main paper [DS08]: proof of theorem 2 section. □

Preliminaries I

Lemma 3.1

Take $q \in \mathbb{F}_d[x_1, \cdots x_n]$ and q_d be its homogenous part of degree d and a, b are elements of $\mathbb{F}^n$ then

$$q(x + a) - q(x + b) = \partial_{q_d}(x, a - b) + E(x)$$

where $deg(E) \leq d - 2$. In other words, the directional derivative of q_d in direction $a - b$ is given by the homogenous part of degree $d - 1$ in the difference $q(x + a) - q(x + b)$

Proof (sketch):

- Note that it is enough to prove the lemma for the case q is a monomial of degree d and then the result follows from linearity and from the fact that derivatives of all monomials in q are of degree smaller than d at most $d - 2$.

Preliminaries II

- Then taking $M(x) = \prod_i x_i^{c_i}$ and observing

$$M(x+a) = M(x) + \sum_i a_i \cdot \frac{\partial M}{\partial x_i}(x) + E_1(x) \quad \text{with } deg(E_1) \leq d-2$$

- And then considering $M(x+b) - M(x+a)$ the result follows.

Preliminaries III

Lemma 3.2

Take $q \in \mathbb{F}_d[x_1, \cdots x_n]$. Given the vector of partial derivatives $\Delta_q(x)$, it is possible to reconstruct q in polynomial time.

Proof (sketch):

- The idea is to go over all monomials of degree $\leq d$ and find out the coefficients they have in q as: For every monomial M, take i the first index such that x_i appears in the M with positive degree.
- Consider $\frac{\partial q}{\partial x_i}(x)$ and check whether the coefficient of the derivative of that monomial is zero or not.
- To get the coefficient in q we divide that by the degree of x_i in the monomial.

Preliminaries IV

Lemma 3.3

C is an $[m, n, k]$ code over $\mathbb{F}$ such that C has an efficient decoding algorithm that can correct an α-fraction of errors. For $i \in [m]$ suppose $a_i \in \mathbb{F}^n$ be the ith row of the generating matrix of C. Then,

1. *$S^0 = (a_1, \cdots, a_m, \overline{0}, \cdots \overline{0}) \in (\mathbb{F}^n)^{2m}$. Then S^0 is an $(n, 1, \alpha/2)$-NIS with an efficient interpolation algorithm.*

2. *$S = (a_1, \cdots, a_m) \in (\mathbb{F}^n)^m$ and suppose that the maximal hamming weight of a codeword in C is smaller than $(1 - 2\alpha) \cdot m$. Then S is an $(n, 1, \alpha)$-NIS with an efficient interpolation algorithm.*

Proof (sketch):

1. - The idea is to first take a degree of one polynomial q. We describe the interpolation algorithm for S^0.

Preliminaries V

- First look at the values of $q(x)$ on the last m points(the zeroes). The majority of these values will be $q(0)$ which will give us the constant term in q.
- Take $q_1(x) = q(x) - q(0)$ is the linear part of q and this reduces to the problem of recovering the homogeneous linear function q_1 form its values on $(a_1, \cdots, a_m)$ with at most $\alpha \cdot m$ errors.
- This task is achieved using the decoding algorithm for C, since the vector $(q_1(a_1), \cdots, q(a_m))$ is just the encoding of the vector of coefficients of q_1 with the code C.

2

- In this take $(v_1, \cdots v_m) \in \mathbb{F}$ be the list of input values given ($v_i = q(a_i)$ for a $1 - \alpha$ fraction of the is).
- Then we go over all $p = |\mathbb{F}|$ possible choices for $q(0)$ and for each "guess" $c \in \mathbb{F}$ do: Subtract c from the values $(v_1, \cdots v_m)$ and then use the decoding algorithm of C to decode the vector $V_c = (v_1 - c, \cdots, v_m - c)$.

Preliminaries VI

- Clearly, for $c = q(0)$ this procedure will give the list of coefficients (without constant terms) of $q(x)$ as output.
- So we are just required to find which invocation of the decoding algorithm is the correct one.
- Say the decoding algorithm, on input V_c, returns a linear polynomial $q_c(x)$ (there is no constant term).
- We can then check to see whether $q_c(a_i) + c$ is indeed equal to v_i for a $1 - \alpha$ fraction of the is.
- If we can show that this test succeeds only for a single $c \in \mathbb{F}$ then we are done and the lemma shall follow.
- So to prove that, suppose on the contrary that there are two linear polynomials $q_c(x)$ and $q_{c'}(x)$ such that both agree with a fraction $1 - \alpha$ of the input values.

Preliminaries VII

- This mean s that there exist two codeword $W_c, W_{c'} \in \mathbb{F}^m$ in C such that $dist(V_c, W_c) \leq \alpha \cdot m$ and $dist(V_{c'}, W_{c'}) \leq \alpha \cdot m$ which implies that

$$dist(V_c - V_{c'}, W_c - W_{c'}) \leq 2\alpha \cdot m$$

- Now the vectors $V_c - V_{c'}$ has the value $c' - c \neq 0$ in all of its coordinates and so we get that the hamming weight of the codeword $W_c - W_{c'}$ is at least $(1 - 2\alpha) \cdot m$ contradicting the properties of C.

Proof of Main theorem

Proof of Theorem 2.1 I

- Take $S_1 = (a_1, \cdots a_m) \in (\mathbb{F}^n)^m$ be an $(n, 1, \epsilon_1)$-NIS of size $|S_1| = m$ and $S_{d-1} = (b_1, \cdots b_r) \in (\mathbb{F}^n)^r$ be an $(n, d-1, \epsilon_{d-1})$-NIS of size $|S_{d-1}| = r$ and A_{d-1} and A_1 be interpolation algorithms for S_{d_1} and S_1 respectively.

- Take $S_d = S_{d-1} \boxplus S_1$. We shall prove the theorem 2.1 by showing that S_d has an interpolation algorithm that makes at most a polynomial number of calls to A_{d-1} and to A_1 and can "correct" a fraction
$$\epsilon_d = \frac{\epsilon_1 \cdot \epsilon_{d-1}}{2}$$
of errors.

- We shall describe the algorithm A_d and prove its correctness simultaneously (the number of calls to A_{d-1} and A_1 will clearly be polynomial as we shall see).

Proof of Theorem 2.1 II

- So fix, $q \in \mathbb{F}_d[x_1, \cdots x_n]$ to be some degree d polynomial and q_d be its homogeneous part of degree d.

- Now denote $S_d = (c_1, \cdots c_{mr})$ where each $c_i \in \mathbb{F}^n$. We also denote by $e = (e_1, \cdots e_{mr}) \in \mathbb{F}^{mr}$ the list of "errors", so that $|\{i \in [mr] | e \neq 0\}| \leq \epsilon_d \cdot mr$.

- The list S_d can be partitioned in a natural way into all the "shifts" of the list S_{d-1}. Now define for each $i \in [m]$ the list $T_i = (b_1 + a_i, \cdots b_r + a_i) \in (\mathbb{F}^n)^r$. We thus have that S_d is the concatenation of $T_1, \cdots T_m$.

- We can also partition the list of errors in a similar way into m lists $e^{(1)}, \cdots e^{(m)}$, each of length r such that $e^{(i)}$ is the list of errors corresponding to the points in T_i.

Proof of Theorem 2.1 III

- We say that an index $i \in [m]$ is *good* if the fraction of errors in T_i is at most $\epsilon_{d-1}/2$, otherwise we say that i is *bad*. In other words, i is good if

$$\left|\left\{j \in [r] | e_j^{(i)} \neq 0\right\}\right| \leq (\epsilon_{d-1}/2) \cdot |T_i| = (\epsilon_{d-1}/2) \cdot r$$

- Now take $E = \{i \in [m] | i \text{ is bad}\}$. From the bound on the total number of errors we get that

$$|E| \leq \epsilon_1 \cdot m$$

The algorithm is divided into three steps.

- The idea is, in the first step we look at all pairs (T_i, T_j) and from each one attempt to reconstruct, using A_{d-1}, the directional derivative $\partial_{q_d}(x, a_i - a_j)$. We will claim that this step gives the correct output for most pairs (T_i, T_j).

Proof of Theorem 2.1 IV

- In the next step we take all the directional derivatives obtained in the first step and from them reconstruct, using A_1, the vector of derivatives $\Delta_{q_d}(x)$ and so also recover $q_d(x)$.

- In the last step of the algorithm we recover the polynomial $q_{\leq d-1}(x) = q(x) - q_d(x)$, again using A_{d-1} which shall give us

$$q(x) = q_{\leq d-1}(x) + q_d(x)$$

Proof of Theorem 2.1 V

The algorithm

Step 1

- Take $i \neq j \in [m]$ be two good indices as defined above, we shall show how to reconstruct $\partial_{q_d}(x, a_i - a_j)$ from the values in T_i, T_j.
- Recall that we have two lists of values

$$L_i = \left(q(b_1 + a_i) + e_1^{(i)}, \cdots q(b_r + a_i) + e_r^{(i)}\right)$$

and

$$L_j = \left(q(b_1 + a_j) + e_1^{(j)}, \cdots q(b_r + a_j) + e_r^{(j)}\right)$$

- Now taking the the differences we get that

$$L_{ij} = L_i - L_j = \Big(q(b_1 + a_i) - q(b_1 + a_j) + e_1^{(i)} - e_1^{(j)},$$
$$\cdots, q(b_r + a_i) - q(b_r + a_j) + e_r^{(i)} - e_r^{(j)}\Big)$$

Proof of Theorem 2.1 VI

and observe that since i and j are both good we have that the terms $e_l^{(i)}$ and $e_l^{(j)}$ is non zero for at most $\epsilon_{d-1} \cdot r$ values of $l \in [r]$.

- Therefore, we can use algorithm A_{d-1} to recover the degree $d-1$ polynomials $Q_{ij}(x) := q(x + a_i) - q(x + a_j)$ from the list L_{ij}.
- From lemma 3.1 we see that throwing away all monomials of degree less than $d-1$ in Q_{ij} leaves us with $\partial_{q_d}(x, a_i - a_j)$.
- On carrying out the first step for all pairs (T_i, T_j) and obtaining $\binom{m}{2}$ homogeneous polynomials of degree $d-1$, we denote them by $R_{ij}(x)$.
- Now since we know that i and j are both good we get

$$R_{ij}(x) = \partial_{q_d}(x, a_i - a_j)$$

If either i or j is bad, we do not know anything about $R_{ij}(x)$.

Proof of Theorem 2.1 VII

Step 2

- In this step we take the polynomials R_{ij} obtained in the first step and recover from them the polynomials $\Delta_{q_d}(x)$ (after which using lemma 3.2 shall give us $q_d(x)$).
- We start by giving some notations: The set of degree $d-1$ monomials is indexed by the set

$$I_{d-1} = \{(\alpha_1, \cdots, \alpha_n) | \alpha_i \geq 0, \alpha_1 + \cdots + \alpha_n = d-1\}$$

- We denote $x^\alpha = \prod_i x_i^{\alpha_i}$ and $coef(x^\alpha, h)$ the coefficient of the monomial x^α in a polynomial $h(x)$. Take $\alpha \in I_{d-1}$ and define the degree 1 polynomial

$$U_\alpha(y_1, \cdots y_n) = \sum_{l=1}^{n} coef\left(x^\alpha, \frac{\partial q_d}{\partial x_l}\right) y_l$$

Proof of Theorem 2.1 VIII

- Now observe that

$$\begin{aligned} \partial_{q_d}(x, a_i - a_j) &= \sum_{l=1}^{n} (a_i - a_j)_l \cdot \frac{\partial q_d}{\partial x_l}(x) \\ &= \sum_{\alpha \in I_{d-1}} x^\alpha \cdot U_\alpha(a_i - a_j) \end{aligned} \tag{1}$$

- Therefore, for each pair i, j such that i and j are good we can get the (correct) values $U_\alpha(a_i - a_j)$ for all $\alpha \in I_{d_1}$ by observing the coefficients of R_{ij}.

- Now fix some $\alpha \in I_{d-1}$ and using the procedure implied above for all pairs $i \neq j \in [m]$, we get $\binom{m}{2}$ values $u_{ij} \in \mathbb{F}$ such that if i and j are good then

$$u_{ij} = U_\alpha(a_i - a_j)$$

- We now recover U_α from u_{ij}s. Repeating this procedure for all $\alpha \in I_{d-1}$ shall give $\Delta_{q_d}(x)$.

Proof of Theorem 2.1 IX

- Since α is fixed apriori, we denote $U(y) = U_\alpha(y)$. We have a list of values $(u_{ij})_{i,j\in[m]}$ such that there exists a set $E = \{i \in [m] | i \text{ is bad}\}$of size $|E| \leq \epsilon_1 \cdot m$ such that if i and j are not in E then $u_{ij} = U(a_i - a_j)$.
- Now partition the list (u_{ij}) according to the index j into disjoint lists: $B_j = (u_{1j}, \cdots, u_{mj})$. If $j \notin R$ then the list B_j contains the values of the degree 1 polynomials $U_j(y) = U(y - a_j)$ on the set S_1 with at most $\epsilon_1 \cdot m$ errors (that is the errors will correspond to indices $i \in E$).
- Therefore, we can use A_1 in order to reconstruct U_j, and from it U. But now the "problem" is that we do not know which js are good.
- This problem can be solved by applying the above procedure for all $j \in [m]$ and then taking the majority vote. Since all good js will return the correct $U(y)$ we will have a clear majority of at least a $1 - \epsilon_1$ fraction.
- Combining all of the above gives us the polynomial $q_d(x)$ and thus completing this step.

Proof of Theorem 2.1 X

Step 3

- Now we have recovered q_ds from the previous steps, so we now abstract out the value $q_d(c_i)$ from the input list of values (which are values of $q(x)$ on S_d, with ϵ_d fraction of errors "noise").
- This reduces us to the problem of recovering the degree $d-1$ polynomial $q_{\leq d-1} = q(x) - q_d(x)$ from its values in S_d with a fraction ϵ_d of errors.
- But this can be solved by using the algorithm A_{d-1} (17) on the values in each list T_j and then taking the majority.
- Since for good js T_j contains at most $\epsilon_{d-1} \cdot r$ errors, and since there are more than half good js, we will get a clear majority and so be able to recover $q_{\leq d-1}$.

Proof of corollary 2.1 I

- In order to prove corollary 2.1, we just need to construct, for all n an $(n, 1, \epsilon)$-NIS $S_1^{(n)}$ with an efficient interpolation algorithm and ϵ which does not depend on n. The corollary will then follow using theorem 2.1.

- To construct S_1 we take a good collection of linear codes $\{C_n\}$, where C_n is an $[m_n, n, k_n]$-code over $\mathbb{F}$ that has an efficient decoding algorithm that can decode a constant fraction of errors and such that the generating matrix of C_n is found in polynomial time (which are known to exist from the result in [MS77]).

Proof of corollary 2.1 II

- Now take $a_1 \cdots a_{m_n} \in \mathbb{F}^n$ be the rows of its generating matrix. We define $S_1^{(n)}$ to be the list of points $(a_1, \cdots a_{m_n}, b_1, \cdots, b_{m_n})$, where for each $j \in [m_n]$ we set $b_j = 0$.
- That is, $S_1^{(n)}$ contains the rows of the generating matrix of a good code, together with the points 0, taken with multiplicity m_n. Lemma 3.3 now shows that $S_1^{(n)}$ satisfies the required conditions.

References

[DS08] Zeev Dvir and Amir Shpilka. "Noisy interpolating sets for low degree polynomials". In: *2008 23rd Annual IEEE Conference on Computational Complexity*. IEEE. 2008, pp. 140–148.

[MS77] Florence Jessie MacWilliams and Neil James Alexander Sloane. *The theory of error-correcting codes*. Vol. 16. Elsevier, 1977.

Thank you!

$$G = (\mathcal{V}_1 \cup \mathcal{V}_2,\ \mathcal{T}_1 \cup \mathcal{T}_2, \mathcal{P}, S_2),$$