CSS.413.1 Topics in Coding Theory

Instructor: Mrinal Kumar

TIFR 2025, Aug-Nov

Scribe: Soham Chatterjee

soham.chatterjee@tifr.res.in
Website: sohamch08.github.io

Contents

1 Targets

The content of this course will be the followings:

- Introduction to Coding Theory: Definitions, Basic Properties, Linear Codes
- Reed Solomon Codes, Reed Muller Codes
- Decoding algorithms for Reed Solomon Codes:
  - Barlekamp-Welch Algorithm
  - Sudan's List Decoding Algorithm
  - Guruswami-Sudan List Decoding Algorithm upto the Johnson Bound
- Univariate Multiplicity Codes – Decoding upto the List Decoding Capacity
- Bounds on the list size
- Local Decoding (LDC), Local Correction (LCC) of Codes
- Local Correction of Reed Muller Codes
- High Variate Locally correctable/decodable codes
- Local Decoding with constant queries – Matching Vector Codes
- Private Information Retrieval – Definitions, constructions
- Lower Bounds for LDCs – Lower Bound for 2-query/4-query/Kalz-Trevisan/Alrabiah-Guruswami
- Local Testing of Codes:
  - Low-Degree Testing
  - Polischuk-Speilman Test
  - Friedl-Sudan Test
  - Arora-Sudan Test
  - Raz-Safra Test
- Applications: Explicit constructions
  - Combinatorial Designs
  - Subspace Designs
  - Derandomization
  - Hardness vs Randomness

2 Basics of Coding Theory

3 Decoding of Reed-Solomon Codes

4 Locally Decodable Codes and Locally Correctable Codes

5 Local Correction of Reed-Müller Codes

6 Multiplicity Codes

Multiplicity codes are a family of recently-introduced algebraic error-correcting codes based on evaluations of polynomials and their derivatives. Specifically, a codeword of a multiplicity code is obtained by evaluating a polynomial of degree at most k, along with all its derivatives of order $< s$, at n points of a finite field $\mathbb{F}_q^m$. These codes were introduced by Kopparty, Saraf and Yekhanin in [KSY14]. Notice that when $s = 1$ this is basically the Reed-Solomon code when $m = 1$ and Reed-Muller code when $m > 1$.

6.1 Construction

Let $s, k, m \in \mathbb{Z}_0$ and let q be a prime power. Let $\Sigma = \mathbb{F}_q^{\binom{s+m-1}{m}}$. For $P(X_1, \ldots, X_m) \in \mathbb{F}_q[X_1, \ldots, X_m]$ we define the order s evaluations of P at $\mathbf{a} \in \mathbb{F}_q$ to be the vector $(P^{(\mathbf{i})}(\mathbf{a}))_{w(\mathbf{i}<s)} \in \Sigma$ where $wt(\mathbf{i}) = \sum_{j=1}^{m} i_j$. Let E be a subset of n points in $\mathbb{F}_q^m$.

Definition 6.1.1: Multiplicity Codes

The multiplicity code of order-s evaluations of degree k polynomials in m variables over all points in E^m is the code over alphabet Σ, and has length n and for each polynomial $P(\mathbf{X}) \in \mathbb{F}_q[\mathbf{X}]$ with $\deg(P) \le k$ the corresponding codeword is

$$\mathrm{Enc}_{s,k,m,q}(P) = (P^{(<s)}(\mathbf{a}))_{\mathbf{a}\in E} \in \Sigma^{n^m}$$

Our current interest is in the case $m = 1$. So

$$\mathrm{Enc}_{s,k,1}(P) = \left(\begin{bmatrix} f(a_1) \\ f^{(1)}(a_1) \\ \vdots \\ f^{(s-1)}(a_1) \end{bmatrix}, \begin{bmatrix} f(a_2) \\ f^{(1)}(a_2) \\ \vdots \\ f^{(s-1)}(a_2) \end{bmatrix}, \cdots, \begin{bmatrix} f(a_n) \\ f^{(1)}(a_n) \\ \vdots \\ f^{(s-1)}(a_n) \end{bmatrix} \right)$$

Remark: The above encoding in not the encoding

$$\mathrm{Enc}_{s,k,1} = \left(f(a_1), f^{(1)}(a_1), \cdots, f^{(s-1)}(a_1), f(a_2), f^{(1)}(a_2), \cdots, f^{(s-1)}(a_2), \cdots, f(a_n), f^{(1)}(a_n), \cdots, f^{(s-1)}(a_n) \right)$$

Each alphabet of the codeword is a vector of size s. The same holds for the multivariate case

The above operation of treating a vector as a single alphabet is called *folding*.

6.2 Rate and Distance of Multiplicity Codes

We will now calculate the rate and the distance of the code. The block length is n^m. Since we are evaluating all the derivatives of order $< s$, the alphabet size is $q^{\binom{s+m-1}{m}}$. So the number of codewords is $\left(q^{\binom{s+m-1}{m}}\right)^{n^m} = q^{n^m\binom{s+m-1}{m}}$. The number of polynomials in m variables of degree at most k is $q^{\binom{k+m}{m}}$. So the rate of the code is

$$R = \frac{\binom{k+m}{m}}{n^m\binom{s+m-1}{m}} \approx \left(\frac{k}{ns}\right)^m$$

Now using the Multiplicity Schwartz-Zippel lemma we can calculate the distance of the code. We have the relative distance to be $\delta = 1 - \frac{k}{ns}$.

Theorem 6.2.1

The rate and the distance of the multiplicity code are $R = \frac{\binom{k+m}{m}}{n^m\binom{s+m-1}{m}} \approx \left(\frac{k}{ns}\right)^m$ and $\delta = 1 - \frac{k}{ns}$ respectively.

We usually think m and s to be large constant. So as multiplicity code achieves the Singleton bound asymptotically.

6.3 List Decoding of Univariate Multiplicity Codes up to Capacity

Since we are interested in univariate multiplicity codes, we will set $m = 1$. So we have three parameters k, s, n and the field size q. Therefore, as we have calculated before the rate and distance of the univariate multiplicity code are $R = \frac{k+1}{ns} \approx \frac{k}{sn}$ and $\delta = 1 - \frac{k}{ns}$ respectively.

6.3.1 Polynomial List Size up to Capacity

Theorem 6.3.1 [Kop15, GW11]
For every $\epsilon \in (0,1)$, there exists $s_0 \approx \frac{1}{\epsilon^2}$ such that the univariate multiplicity code with multiplicity parameter $s > s_0$ can be efficiently list decodable from $\left(1 - \frac{k}{ns} - \epsilon\right)$ fraction of errors.

We will give the proof in [GW11]. It uses polynomial method based arguments. This proof has two steps.

Step 1: Interpolation

Step 2: Reconstruction of close enough codewords

So assume the received word is $w = \left(\alpha_0, \beta_{i,0}, \beta_{i,1}, \ldots, \beta_{i,s-1}\right)_{i=1}^{n}$. With this we will show the proof of the above theorem.
Proof: First we will set some parameters. Let $t = \sqrt{s} \approx \frac{1}{\epsilon}$.

Step 1: Interpolation

In step 1 we will look for an $t+1$ variate polynomial $Q(X, Y_1, \ldots, Y_t)$ which is linear in Y_i's i.e.

$$Q(X, Y_1, \ldots, Y_t) = A_0(X) + A_1(X)Y_1 + \cdots + A_t(X)Y_t$$

Let f is a close enough polynomial. Then define $R_f(X) = Q(X, f(X), f^{(1)}(X), \ldots, f^{(t-1)}(X))$. Then we want $R_f(X) \equiv 0$. And also we want whenever f and the received word agree on some point $R_f(X)$ has a zero of high multiplicity at that point. So let f agrees with the received word at α. Then

$$R_f(\alpha_i) = Q(\alpha_i, f(\alpha_i), f^{(1)}(\alpha_i), \ldots, f^{(t-1)}(\alpha_i)) = Q(\alpha_i, \beta_{i,0}, \beta_{i,1}, \ldots, \beta_{i,t-1}) = 0$$

Now

$$R_f^{(1)}(X) = \frac{dA_0}{dX}(X) + \frac{d}{dX}\left(\sum_{i=1}^{t} A_i(X) f^{(i-1)}(X)\right) = \frac{dA_0}{dX}(X) + \sum_{i=1}^{t} \frac{dA_i}{dX}(X) \cdot f^{(i-1)}(X) + A_i(X) \cdot f^{(i)}(X)$$

Therefore

$$R_f^{(1)}(\alpha_i) = \frac{dA_0}{dX}(\alpha_0) + \sum_{j=1}^{t} \frac{dA_j}{dX}(\alpha_i) \cdot \beta_{i,j-1} + A_j(\alpha) \cdot \beta_{i,j}$$

So we want as many derivatives of R_f to be zero as possible.

Observation 1. *In $R_f^{(k)}(X)$ we needed the evaluations of $\beta_{i,0}, \ldots, \beta_{i,t-1}, \beta_{i,t}, \ldots, \beta_{i,t+k-1}$.*

Since we have evaluations till $(s-1)^{th}$ order derivative we can only take derivative of R_f upto order $(s-t)$. So we want $R_f^{(k)}(\alpha_i) \equiv 0$ for all $k \in \{0, \ldots, s-t\}$. And Q follows the following properties:

- $\deg(A_i) \leq D$
- For all $i \in [n]$, $R_f^{(k)}(\alpha_i) \equiv 0$ for all $k \in \{0, \ldots, s-t\}$. To make it simple define the operator Ψ as

$$\Psi(Q) := A_0^{(1)}(X) + \sum_{i=0}^{t} (A_i^{(1)}(X)Y_i + A_i(X)Y_{i+1})$$

and $\Psi^i(Q) = \Psi(\Psi^{i-1}(Q))$, $\Psi^0(Q) = Q$. Then $\forall\, i \in [n], \forall\, j \in \{0, \ldots, s-t\}$, $\Psi^j(Q)(\alpha, \overline{\beta}_i) = 0$.

Observation 2. *Each point of agreement of f is a root of R_f of multiplicity at least $s - t + 1$.*

Now for step 1 to return a Q successfully we need the number of variables to be more than the number of equations. The number of variables is $(t+1)(D+1)$. The number of equations is $n(s-t+1)$. So we need

$$(t+1)(D+1) > n(s-t+1) \iff D+1 > \frac{n(s-t+1)}{t+1}$$

Hence enough to take $D = \frac{n(s-t+1)}{t+1}$. Then step 1 returns a nonzero Q.

Observation 3. *If f has agreement $> \frac{D+k}{s-t+1}$ with the received word then $R_f(X) \equiv 0$ as $\deg(R_f) \leq D+k$ and each point of agreement is a zero of multiplicity at least $s-t+1$.*

So the number of agreements is more than

$$\frac{D+k}{s-t+1} = \frac{\frac{n(s-t+1)}{t+1}+k}{s-t+1} = \frac{n}{t+1} + \frac{k}{s}\cdot\frac{s}{s-t+1} \approx \epsilon n + \frac{k}{s}$$

since we take s to be constant.

Step 2: Reconstruction of close enough codewords
Find all degree k, $f(X)$ such that

Step 2.1: $Q(X, f(X), f^{(1)}(X), \dots, f^{(t-1)}(X)) \equiv 0$ [This step looks like solving a differential equation]

Step 2.2: f has large agreement with the received word.

For the step 2.1 let $f_1, \dots, f_l$ are the solutions. Then any linear combination of them is also a solution. Hence the space of solutions of f is a vector space over the field. We need to argue that the dimension of this space is at small. Let S be the set of all $f \in \mathbb{F}_q[X]$ such that $\deg(f) \leq k$ and $Q(X, \overline{f}(X)) \equiv 0$. Then by Lemma 6.3.2 we have $\dim S \leq t-1$. Since s is constant, t is also constant. So the number of solutions is at most q^{t-1}. Hence the list size is at most $q^{t-1} = \text{poly}(n)$. ■

Lemma 6.3.2
S is a subspace of dimension at most $t-1$.

Proof: WLOG we can assume $A_i(0) \neq 0$ for all $i \in \{0, \dots, t\}$ otherwise we can do a random shift to make it nonzero. Let $f \in S$, then

$$Q(X, f(X), f^{(1)}(X), \dots, f^{(t-1)}(X)) = A_0(X) + A_1(X)\cdot f(X) + A_2(X) f^{(1)}(X) + \cdots + A_t(X)\cdot f^{(t-1)}(X) \equiv 0$$

Let $A_i(X) = \sum\limits_{j=0}^{D} A_{i,j} X^j$ for all $i \in \{0, \dots, t\}$ and $f(X) = \sum\limits_{j=0}^{k} f_j X^j$. Therefore

$$f^{(i)}(X) = \sum_{j=0}^{k-i} \frac{(i+j)!}{j!} f_{i+j} X^j$$

Then the coefficient of X^i in $Q(X, f(X), f^{(1)}(X), \dots, f^{(t-1)}(X))$ is

$$\begin{aligned} &A_{0,i} + \left(\sum_{j=0}^{i} A_{1,j}\cdot f_{i-j}\right) + \left(\sum_{j=0}^{i} A_{2,j}\cdot (i+1-j) f_{i+1-j}\right) + \cdots + \left(\sum_{j=0}^{i} A_{t,j}\cdot \frac{(t-1+i-j)!}{(i-j)!} f_{t-1+i-j}\right) \\ =&A_{0,i} + \sum_{l=1}^{t}\sum_{j=0}^{i} A_{l,i-j}\cdot \frac{(l-1+j)!}{j!} f_{l-1+j} \end{aligned}$$

Since f is a solution this coefficient is 0. Notice that in the above linear equation coefficient of X^i depends on f_j for all $j < i+t$. Hence we can determie f_{i+t-1} uniquely if we have $f_0, \dots, f_{i+t-2}$ by using the coefficient of X^i to be zero. The coefficient of X^0 needs $f_0, \dots, f_{t-1}$. So once we fix $f_0, \dots, f_{t-2}$ we can determine uniquely all the other coefficients. Hence the dimension of S is at most $t-1$. ■

6.3.2 Constant List Size up to Capacity

Now we will show that not only the list size is polynomial but it is actually constant, more precisely it only depends on ϵ.

Theorem 6.3.3 [KRZSW18]
The list size above is of constant size only depends on ϵ and independent of the block length.

We will basically show that if you take a subspace of small dimension then there can't be too many codewords in that subspace which are close enough to the received word. So we will use the following idea:

Idea. *Give a randomized algorithm and show that every close enough polynomial in S is output by the algorithm with probability at least some constant.*

Proof: Consider the following randomized algorithm:

Algorithm 1: Randomized List Decoder

```
1 Pick i_1, ..., i_{t-1} coordinates independently uniformly at random;
2 if ∃! f ∈ S such that f agrees with w at i_1, ..., i_{t-1} then
3     return f
```

Observation 4. *Number of coordinates is the same as the subspace dimension since after fixing the values at those coordinates the polynomial is fixed.*

Claim 6.3.4
Fix any $f \in S$ such that f agrees with w on at least $\frac{k}{s} + \epsilon n$ coordinates. Then the probability that the above algorithm returns f is at least ϵ^{t-1}.

Proof: The above algorithm outputs f if f agrees with w at those $(t-1)$ points and every other polynomials $\hat{f} \in S$ disagrees with f in at least one of those $(t-1)$ points. Let S_j be the subspace of polynomials in S which agrees with w at i_l^{th} coordinate for all $l \in [j]$. Then for the algorithm to output f we need $\dim S_{j+1} < \dim S_j$ for all $j \in [t-1]$, take $S_0 = S$ and $f \in S_{t-1}$. Let E_j be the event that $\dim S_j < \dim S_{j-1}$ and $f \in S_j$.

Observation 5. *If $\bigwedge_{j=1}^{t-1} E_t$ happens then the algorithm outputs f.*

$$\mathbb{P}\left[\bigwedge_{j=1}^{t-1} E_j\right] = \prod_{j=1}^{t-1} \mathbb{P}\left[E_j \;\middle|\; \bigwedge_{l=1}^{j-1} E_l\right]$$

Now assume $\bigwedge_{l=1}^{j-1} E_l$. We have S_l containing f and $\dim S_l = t-1-l$. To calculate $\mathbb{P}\left[E_{l+1} \;\middle|\; \bigwedge_{l=1}^{j} E_l\right]$ it is enough to show that there exists an $h \in S_j$ such that f and h disagree at i_{j+1} and f, w agrees at i_{j+1}. Therefore

$$\mathbb{P}\left[E_{l+1} \;\middle|\; \bigwedge_{l=1}^{j} E_l\right] = \mathbb{P}_{i_{j+1}} = [f, w \text{ agree at } i_{j+1} \wedge \exists\, h \in S_j \text{ such that } f_{i_{j+1}} \neq h_{i_{j+1}}]$$

Now f and w agrees in at least $\frac{k}{s} + \epsilon n$ coordinates. And by Theorem 6.2.1 f and any other codeword of S can agree in at most $\frac{k}{s}$ coordinates. Fix any other codeword $h \in S_j$, $h \neq f$. So there are at least ϵn coordinates where f and w agrees but f and h disagrees. Therefore

$$\mathbb{P}_{i_{j+1}} = [f, w \text{ agree at } i_{j+1} \wedge \exists\, h \in S_j \text{ such that } f_{i_{j+1}} \neq h_{i_{j+1}}] \geq \epsilon$$

Hence the algorithm outputs f with probability at least ϵ^{t-1}. ■

The above claim now directly implies that the list size is at most $\frac{1}{\epsilon^{t-1}}$. The algorithm outputs any close enough polynomial with probability at least ϵ^{t-1}. So if there are l such polynomials then the sum of the probabilities that the algorithm outputs those polynomials is at most 1. Hence $l \leq \epsilon^{-(t-1)}$. Since ϵ and t are constants this proves that the list size is constant. ■

6.4 Local Correction of Multiplicity Codes

7 Matching Vector Codes

Here we give a detailed treatment of locally decodable codes that arise from families of matching vectors. Any construction of such codes naturally falls into two parts: the design of a matching vector family and the actual code construction. First we will discuss how to locally decode, and then we will talk about matching vector families. We will show the construction of 3-query LDCs.

7.1 Construction

7.2 Local Decoding

8 References

[GW11] Venkatesan Guruswami and Carol Wang. *Optimal Rate List Decoding via Derivative Codes*, pages 593–604. Springer Berlin Heidelberg, 2011.

[Kop15] Swastik Kopparty. List-Decoding Multiplicity Codes. *Theory of Computing*, 11(1):149–182, 2015.

[KRZSW18] Swastik Kopparty, Noga Ron-Zewi, Shubhangi Saraf, and Mary Wootters. Improved Decoding of Folded Reed-Solomon and Multiplicity Codes. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 212–223. IEEE, October 2018.

[KSY14] Swastik Kopparty, Shubhangi Saraf, and Sergey Yekhanin. High-rate codes with sublinear-time decoding. *Journal of the ACM*, 61(5):1–20, September 2014.