CS 229r Essential Coding Theory Jan 27, 2020

Lecture 6

Instructor: Madhu Sudan *Scribe: Luowen Qian*

We will start by reviewing fields and polynomials from algebra. The scribe notes will contain a succinct presentation of concepts following the presentation in the lecture. For a more extensive reference, the readers are encouraged to read *Lecture Notes on Algebra* posted by Madhu Sudan.

1 Commutative Rings and Fields

Fields are mathematical objects modeled by rationals $\mathbb{Q}$. Somewhat formally, a commutative ring is a set R with the following operations:

- Addition (denoted by $a + b$) that is associative, commutative, and invertible (denoted by $-a$) with respect to additive identity (denoted by 0);
- Multiplication (denoted by $a \cdot b$, sometimes simply ab) that is associative, commutative, has multiplicative identity (denoted by 1), and satisfies distributivity $a(b + c) = ab + ac$.

Furthermore, if the multiplication is invertible (denoted by a^{-1}) except for 0, $(R, +, \cdot)$ (sometimes simply R if the operations are apparent from the context) is a field.

Theorem 1 (Finite fields)**.** *For any $q = p^t$ where p is a prime, and t is a positive integer, there exists a unique*[1] *finite field $\mathbb{F}_q$ of size*[2] *q. Furthermore, a field of size q exists only if q is a prime power.*

Fact 2 (Prime fields)**.** *For $t = 1$, or equivalently if $q = p$ is a prime, the field $\mathbb{F}_q$ can be simply represented as $\mathbb{Z}_q$ with operations given by addition and multiplication modulo q.*

2 Polynomial Rings

Given any commutative ring R, we denote $R[x]$ to be the set of all polynomials with coefficients from a field R over a formal variable x with naturally defined addition and multiplication operations. Somewhat more formally, each non-zero element f (or polynomial $f(x)$) in $R[x]$ is given by a finite sequence $f = \langle f_0, ..., f_d \rangle$ where each $f_i \in R$ and $f_d \neq 0$. We define the degree of a non-zero polynomial $\deg f$ to be d and degree of zero is not important.

Proposition 3. *For any commutative ring R, $R[x]$ is a commutative ring.*

For the rest of the lecture, we will consider exclusively polynomials over finite fields.

Example 4. *Consider polynomials over binary fields $\mathbb{F}_2[x]$. $x+1$ and $1+x+x^2$ are elements of the field. We can evaluate the sum of two to x^2, and their product being $(x+1)(1+x+x^2) = x+x^2+x^3+1+x+x^2 = 1+x^3$.*

Definition 5 (Evaluation of a polynomial)**.** $\mathrm{Eval} : \mathbb{F}[x] \times \mathbb{F} \mapsto \mathbb{F}$ *is a function that evaluates a polynomial in the natural way, namely, given polynomial f and element α,* $\mathrm{Eval}(f, \alpha) = \sum_{i=0}^{\deg(f)} f_i \alpha^i$, *and* $\mathrm{Eval}(0, \alpha) = 0$.

Sometimes, we simply write $f(\alpha) := \mathrm{Eval}(f, \alpha)$.

Exercise 6. *Prove that $x^2 = x$ for all $x \in \mathbb{F}_2$ and therefore $1 + x + x^2 = 1$ for all $x \in \mathbb{F}_2$.*

Exercise 7. *Using the exercise above as a hint, prove that under* Eval, *$\mathbb{F}_q[x]$ has only q^2 different polynomials for any finite field $\mathbb{F}_q$. (note that there are q^q different functions of $\mathbb{F}_q \mapsto \mathbb{F}_q$)*

[1]Up to isomorphism.
[2]Also referred to as order.

Proposition 8 (Division algorithm). *Given $f, g \in \mathbb{F}[x]$ and $g \neq 0$, there exists unique $q, r \in \mathbb{F}[x]$ such that $f(x) = q(x)g(x) + r(x)$ with $\deg r < \deg g$.*

Example 9. *If we divide $p(x)$ by $(x - \alpha)$, we get $r(x) = p(\alpha)$. In particular, if α is "root" of p, then $(x - \alpha)$ divides $p(x)$.*

Fact 10. *$\mathbb{F}[x]$ is a unique factorization domain, meaning there exists a unique factorization for every polynomial.*

Combining these two facts, we get the following theorem.

Theorem 11. *If a non-zero polynomial $p \in \mathbb{F}[x]$ has degree at most d, then p has at most d roots (in $\mathbb{F}$).*

3 Reed-Solomon Code

Definition 12. *Given a finite field $\mathbb{F}_q$, integers $0 < k \leq n \leq q$, and n distinct elements $\langle \alpha_1, ..., \alpha_n \rangle \in \binom{\mathbb{F}_q}{n}$. A Reed-Solomon code $RS_{\mathbb{F}_q, n, k, \langle \alpha_1, ..., \alpha_n \rangle}$ is a code given by the following encoding algorithm:*

1. *On input $c_0, ..., c_{k-1}$, let $M(x) \in \mathbb{F}_q[x]$ be a polynomial given by $M(x) = \sum_{i=0}^{k-1} c_i x^i$;*
2. *Output $(M(\alpha_1), ..., M(\alpha_n))$.*

Theorem 13. *Reed-Solomon code $RS_{\mathbb{F}_q, n, k, \langle \alpha_1, ..., \alpha_n \rangle}$ is an $[n, k, n - k + 1]_q$-code.*

Proof. Recall that a $(n, k, d)_q$-code C is a subset of Σ^n with $|\Sigma| = q$ if:

- $|C| = q^k$;
- $\forall x \neq y \in C$, $\Delta(x, y) \geq d$.

A $[n, k, d]_q$-code C is a $(n, k, d)_q$-code satisfying C is a linear subspace of Σ^n, i.e. if $\alpha \in \Sigma, x, y \in C$ then $\alpha x + y \in C$.

To show that Reed-Solomon code is linear, observe that codewords are evaluation of polynomial of degree less than k, and polynomials preserve degree under addition of polynomials and multiplication of polynomial and $\mathbb{F}_q$. An alternative way to show this is by demonstrating the generator matrix, which is given by the Vandermonde matrix where the i-th column is $1, \alpha_i, ..., \alpha_i^{k-1}$.

This code has distance $n - k + 1$ by Theorem 11, namely, if M, N are different polynomials with degree less than k, then the polynomial $M - N$ is non-zero and has degree less than k, and therefore $M - N$ has less than k roots, therefore the equation $M = N$ has at most $k - 1$ solutions. $\square$

As we observed last lecture, the parameter we achieve with Reed-Solomon code is optimal. However, the undesirable property of the code is that it requires a large alphabet.

Decoding Reed-Solomon code. In Reed-Solomon code, the encoder gives out $M(\alpha_1), ..., M(\alpha_n)$, and after some errors the decoder receives $y_1, ..., y_n$. If there are no errors, we can simply solve a linear system to recover $c_0, ..., c_{k-1}$. If errors occur, a naive approach to decoding the code would be to try all possible uncorrupted coordinates and take the majority vote, however, the number of possibilities is $\binom{n}{k} \approx n^k$ so this approach is clearly inefficient. In the later lectures, we will show how to decode this code efficiently even if errors happen.

4 Other Finite Fields

We showed how to construct Reed-Solomon code for any finite field, however, we still do not quite know how to implement operations for finite fields other than prime fields. Now let us direct our attention to finite fields that are not prime fields, namely, we want to construct or build such fields, and we hope to get from finite fields of prime order to other finite fields by using polynomials of finite fields.

To get inspirations, we observe that $\mathbb{Z}$ is not a field since 2 has no multiplicative inverse. However, if we take the closure of $\mathbb{Z}$ under field operations we get $\mathbb{Q}$ which is a field. Similarly, from $\mathbb{F}[x]$, we can define a fraction field $\widetilde{\mathbb{F}[x]}$ where each element is a/b for some $a, b \in \mathbb{F}[x]$, and we can define additions and multiplications in a natural way. This is not the approach we will be taking for this lecture.

Alternatively, we can also restrict $\mathbb{Z}$ to $\mathbb{Z}_p$ to get a field. Similarly, we can also get such thing. First, let us introduce the analogue of prime in polynomial rings.

Definition 14 (Irreducibility)**.** *We call $g \in \mathbb{F}[x]$ to be reducible if $g = h \cdot f$ for non-zero polynomials f, h with non-zero degree, and g to be irreducible if g is not reducible.*

Lemma 15. *For any finite field $\mathbb{F}_q$ and any d, there exists a monic polynomial $g \in \mathbb{F}_q[x]$ of degree d that is irreducible.*

Fact 16. *Given a finite field $\mathbb{F}_q$ and an irreducible polynomial $g \in \mathbb{F}_q[x]$, $\mathbb{F}_q[x]/g(x) \subseteq \mathbb{F}_q[x]$ is a finite field of size q^d, where additions and multiplications are defined naturally modulo $g(x)$.*

Using this fact, we can build fields of size 2^t for any t, for example, of size $2^8 = 256$, which corresponds to "bytes", which allows us to use Reed-Solomon code over bytes.

In general, we are given some integer $n \in \mathbb{Z}$, and we want to find a finite field of size roughly n. A natural solution is to take the closest prime number, however, finding such prime is hard (despite at least average-case polynomial-time in $\log n$), and in practice a randomized procedure is usually employed. Taking the nearest power of two 2^t is much easier, however, in order to construct the field, we need to find an irreducible polynomial in $\mathbb{F}_2[x]$ of degree t[3]. Enumerating all possible polynomial would take too many time since there are roughly $2^t \approx n$ many polynomials. We know that random polynomial is irreducible with probability $1/t$, although this is no better than simply using a prime. Fortunately, there exists however a deterministic algorithm running in time poly(t) to find such polynomials, which we will not cover in this course.

5 Wozencraft Ensemble

As noted before, the undesirable property of Reed-Solomon code is that the field size could be very large. It remains a problem to construct codes with good rates when the alphabet is small, say binary. Using the theory of finite fields, we will show a binary code with rate and relative distance matching the performance of the greedy code (Varshamov Bound).

In particular, we will show a code with rate 1/2 and relative distance $\delta < 1/2$ satisfying $1/2 = R = 1 - H(\delta)$, i.e. we will show a code of $\mathbb{F}_2^k \mapsto \mathbb{F}_2^{2k}$. In the following, it will be easier to consider $\mathbb{F}_{2^k} \mapsto \mathbb{F}_{2^k}^2$ and then translate it back to the map above. We will use a randomized argument to show that such code exist.

For any $\alpha \in \mathbb{F}_{2^k}$, there exists an encoding function $E_\alpha : \mathbb{F}_{2^k} \mapsto \mathbb{F}_{2^k}^2$ given as $m \mapsto (m, \alpha m)$. Let $\tilde{E}_\alpha : \mathbb{F}_2^k \mapsto \mathbb{F}_2^{2k}$ be an arbitrary corresponding encoding function in the original vector space (given by some bijection).

Lemma 17. *For $\alpha \neq \beta \in \mathbb{F}_{2^k}$, $C_\alpha \cap C_\beta = \{\overline{0}\}$, where $\overline{0} = (0, 0)$.*

Proof. Suppose the opposite, then we have $(m_1, \alpha m_1) = (m_2, \beta m_2) \neq (0, 0)$. This implies that $m_1 = m_2$. It cannot be the case that $m_1 = 0$ as that would lead to $(m_1, \alpha m_1) = (0, 0)$. Therefore $\alpha = \beta$, a contradiction. $\square$

[3]We do not need to know the irreducible polynomial to perform addition.

Lemma 18. *There exists some $\alpha \in F_{2^k}$ such that $\Delta(\tilde{E}_\alpha) \geq d$ where d is the largest integer such that* $\text{Vol}(2k, d-1) \leq 2^k$.

Proof. We call α bad if there exists some non-zero message $m \neq 0$ such that $\Delta(\tilde{E}_\alpha(m), \overline{0}) < d$. Note that every non-zero $\tilde{E}_\alpha(m)$ with distance to $\overline{0}$ less than d only eliminates at most one bad code by the previous lemma, in the end we will have at least $2^k - (\text{Vol}(2k, d-1) - 1) \geq 1$ code remains which satisfies the code distance condition. □

Exercise 19. *Give an explicit construction of a linearity preserving bijection such that it will allow us to prove the existence of a linear code with distance d.*

However, this counting argument does not give us a way to find such good code (in particular, an exhaustive search will run in time 4^k). On the other hand, it also shows that for a random code and a random message, it will have very good distances, so we have not ventured far from the original random matrix code.

Exercise 20. *Prove that most α's give distance at least $d - \sqrt{k}$, therefore, a random α will have good distance for all messages with high probability.*

In the next few lectures, we will use ideas like this to get an explicit deterministic efficient construction. One way to do this is to take polynomials of many variables. We will also try things like representing a large field with small alphabets to get great binary codes.

Definition 21 (Maximum Distance Separable Codes)**.** *Let C a $(n, k, d)_q$-code be a maximum distance separable code if $d = n - k + 1$.*

Exercise 22. *If C is a $[n, k, n-k+1]_q$-code, then $C^\perp$ is a maximum distance separable code.*