AM 106/206: Applied Algebra **Madhu Sudan**[1]

Lecture Notes 18

November 8, 2017

1 Error-Correcting Codes

- **Goal:** encode data so that it can be recovered even after much of it has been corrupted.
  - Useful for storage (hard disks, DVDs), communication (cell phone, satellite).
- **Def:** An *code* is an injective mapping $\mathrm{Enc} : \Sigma^k \to \Sigma^n$ for some finite *alphabet* Σ, *message length* k and *block length* n.
- **Def:** For two strings $x, y \in \Sigma^n$, we define their *Hamming distance* to be

$$D(x, y) = \#\{i \in [n] : x_i \neq y_i\}/n.$$

- **Def:** A code Enc is *t-error-correcting* if there is a *decoding function* $\mathrm{Dec} : \Sigma^n \to \Sigma^k$ such that for every message $m \in \Sigma^k$ and every received word $r \in \Sigma^n$ such that $D(r, \mathrm{Enc}(m)) \leq \delta$, we have $\mathrm{Dec}(r) = m$.
- **Example:** repetition code $n = k \cdot \ell$, $\mathrm{Enc}(m) = (m, m, m, \ldots, m)$ is t-error-correcting if $\ell \geq 2t + 1$.
- **Proposition:** A code Enc is *t-error-correcting* if and only if its *minimum distance* $\min_{m \neq m'} D(\mathrm{Enc}(m), \mathrm{Enc}(m'))$ is greater than $2t$.

  Proof:

  Note: the minimum distance only depends on the set of codewords $C = \{Enc(m) : m \in \Sigma^k\} \subseteq \Sigma^n$ and not on how we map elements of Σ^k to Σ^n. Thus people often use the word *error-correcting code* to refer to the set C rather than the function Enc.
- **Goals:** Construct error-correcting codes for arbitrarily large message lengths k and:

[1] These notes are copied mostly verbatim from the lecture notes from the Fall 2010 offering, authored by Prof. Salil Vadhan. I will attempt to update them, but apologies if some references to old dates and contents remain.

1. Maximize the relative decoding distance $\delta = t/n$, or equivalently the relative minimum distance. Ideally, these should be constants independent of k, e.g. $\delta = .1$).
2. Maximize the *rate* $\rho = k/n$ (ideally constant independent of k, e.g. $\rho = .1$).
3. Minimize the *alphabet size* $|\Sigma|$ (ideally constant independent of k, e.g. $\Sigma = \{0, 1\}$).
4. Have efficient (e.g. polynomial time or even linear time) encoding and decoding algorithms. (Note that decoding algorithm in proposition above is *not* efficient in general — may require enumerating all strings at distance at most t from r.)

2 A Bound on Distance

Proposition: If $E : \Sigma^k \to \Sigma^n$ has distance d then $d \leq n - k + 1$.

Proof: Consider the projection function $\pi : \Sigma^n \to \Sigma^{k-1}$ satisfying $\pi(x_1, \ldots, x_n) = (x_1, \ldots, x_{k-1})$. Since $\pi \circ E : \Sigma^k \to \Sigma^{k-1}$ has a smaller range than domain, it can not be injective. So there exists $m \neq m' \in \Sigma^k$ such that $\pi(E(m)) = \pi(E(m'))$ and so the encoding of m and m' agree on the first $k-1$ coordinates. In other words $D(E(m), E(m')) \leq n - (k-1)$ and so $d \leq n - k + 1$.

The above proof seems quite sloppy. Why did we need image Σ^{k-1} when any set of size less that Σ^k would have found a collision. Why project to the first $k-1$ coordinates - maybe some other set of k coordinates would have yielded the same. Why not find some other pair m and m' which might agree also outside the projected set? There seem to be so many avenues for improvement that it seems almost clear the bound above can be improved. If you're convinced, then the rest of the lecture should be a big surprise. Actually the bound is not weak and the Reed-Solomon codes achieve it tightly.

In order to appreciate the codes, and their algorithmic effectiveness, we state a couple of facts that we will cover in detail in a later lecture. Indeed this lecture will motivate those future lectures on "Finite Fields", and "Algorithms for Polynomials".

Proposition: For every prime p and positive integer k a finite field $\mathbb{F}_{p^k}$ of cardinality p^k exists. Such a field can be constructed in expected time $\text{poly}(k, \log p)$ and field operations can be performed with the same running time.

Proposition: Let $\mathbb{F}$ be a finite field.

1. Two polynomials in $\mathbb{F}[x]$ of degree n can be added with $O(n)$ field operations.
2. Two polynomials in $\mathbb{F}[x]$ of degree n can be multiplied with $O(n \text{poly} \log n)$ field operations.
3. Given polynomial $a(x), b(x) \in \mathbb{F}[x]$ of degree n, the quotient $q(x)$ and remainder $r(x)$ such that $a(x) = q(x)b(x) + r(x)$ can be found using $O(n \text{poly} \log n)$ field operations.
4. The GCD of two polynomials $a(x), b(x) \in \mathbb{F}[x]$ can be found using $O(n \text{poly} \log n)$ field operations.
5. Given a polynomial $a(x) \in \mathbb{F}[x]$ of degree n and $\alpha \in \mathbb{F}$ $a(\alpha)$ can be computed with $O(n)$ field operations.
6. **(Multipoint Evaluation)** Given a polynomial $a(x) \in \mathbb{F}[x]$ of degree n and points $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$, the set $\{(\alpha_1, a(\alpha_1)), \ldots, (\alpha_n, a(\alpha_n))\}$ can be computed with $O(n \text{poly} \log n)$ field operations.

7. **(Interpolation)** Given the evaluations of a polynomial $a(x) \in \mathbb{F}[x]$ of degree less than n, i.e., the set $\{(\alpha_1, a(\alpha_1)), \ldots, (\alpha_n, a(\alpha_n))\}$, the coefficients of $a(x)$ can be computed with $O(n\text{poly}\log n)$ field operations.

8. **(Factorization)** Given $a(x) \in \mathbb{F}[x]$ of degree n a factorization of $a(x)$ into irreducibles can be computed in expected time $\text{poly}(n, \log|\mathbb{F}|)$.

9. **(Bivariate Factorization)** Given $a(x,y) \in \mathbb{F}[x,y]$ of (total) degree n a factorization of $a(x)$ into irreducibles can be computed in expected time $\text{poly}(n, \log|\mathbb{F}|)$.

Most of the algorithms above are trivial if we relax the running time to $\text{poly}(n)$. The two exceptions are the factorization algorithms which we may see a glimpse of later in the course. The near-linear running time is also non-trivial in most of the cases above.

3 Reed–Solomon Codes

- **Reed-Solomon Code:** The *q-ary Reed–Solomon code of message length k and blocklength n* is a code $\text{RS} : \mathbb{F}_q^k \to \mathbb{F}_q^n$ with alphabet $\Sigma = \mathbb{F}_q$. We view the message $m = (m_0, \ldots, m_{k-1}) \in \mathbb{F}_q^k$ as coefficients of a polynomial $p_m(x) = \sum_{i=0}^{k-1} m_i x^i$ of degree at most $d = k-1$. The encoding is $\text{RS}(m) = (p_m(\alpha_1), \ldots, p_m(\alpha_n))$ where $\alpha_1, \ldots, \alpha_n$ are fixed distinct elements of $\mathbb{F}_q$. (Thus we need $q \geq n$.)

- **Proposition:** The minimum distance of the Reed-Solomon code is $n-k+1$, and thus it is t-error-correcting for $t = (n-k)/2$.
Proof:

- Thus, taking e.g. $n = 2k$, we have constant rate ($\rho = 1/2$) and constant relative decoding distance ($\delta = t/n = 1/4$). The only downside is the nonconstant alphabet size ($q \geq n$), but this can be improved by combining Reed–Solomon codes with other codes (see PS10).

- **Efficiency of RS Codes:** The encoding algorithm for Reed–Solomon codes is efficient. It just requires evaluating a degree d polynomial at n points, which can be done with $O(nd)$ operations in $\mathbb{F}_q$ using the naive algorithm, and $O(n \log n)$ operations using Fast Fourier Transforms over $\mathbb{F}_q$. Decoding is nontrivial. Given a received word $r \in \mathbb{F}_q^n$, we want to find a message $m \in \mathbb{F}_q^k$ such that $\text{RS}(m) = (p_m(\alpha_1), \ldots, p_m(\alpha_n))$ has distance at most t from $r = (\beta_1, \ldots, \beta_n)$. This amounts to solving the following problem.

- **Noisy Polynomial Interpolation:** Given n pairs $(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n) \in \mathbb{F}_q \times \mathbb{F}_q$ with $\alpha_1, \ldots, \alpha_n$ distinct, we want to find all polynomials p of degree at most $d = k-1$ such that $p(\alpha_i) = \beta_i$ for at least $s = n-t$ values of i.

- **Thm:** The Noisy Polynomial Interpolation problem can be solved in polynomial time if $s > 2\sqrt{dn}$.

In particular, if $n = 9k > 9d$, we can efficiently decode from $t = n - 2\sqrt{kn} = n/3$ errors and still have constant relative rate (namely $\rho = 1/9$). It is known how to remove the factor of 2 in the theorem, which suffices to correct $t = (n-k)/2$ errors, the same as guaranteed (inefficiently) by the minimum distance of the code.

Proof:

Step 1: Find a nonzero *bivariate* polynomial $Q(x, y)$ such that (a) $Q(\alpha_i, \beta_i) = 0$ for all i, and (b) the degree of Q in x is at most $\sqrt{dn}$ and the degree of Q in y is at most $\sqrt{n/d}$.

- We are looking to find the coefficients c_{ij} of

$$Q(x, y) = \sum_{i=0}^{\sqrt{dn}} \sum_{j=0}^{\sqrt{n/d}} c_{ij} x^i y^j.$$

- Each constraint $Q(\alpha_i, \beta_i) = 0$ is a homogeneous linear constraint on the coefficients c_{ij}. Since there are $(\sqrt{dn}+1)(\sqrt{n/d}+1) > n$ coefficients c_{ij} and only n constraints, there exists a nonzero solution and we can find it by Gaussian elimination.

Step 2: Factor Q into irreducible polynomials, look for any factors of the form $y - p(x)$, and output all such p that appear.

- Observe that if $p(x)$ is a polynomial of degree d such that $p(\alpha_i) = \beta_i$ for at least s values of i, then the *univariate* polynomial $S(x) = Q(x, p(x))$ has at least s roots (namely the values of α_i such that $p(\alpha_i) = \beta_i$). The degree of $S(x)$ is at most $\sqrt{dn} + d \cdot \sqrt{n/d} = 2\sqrt{dn}$. Since $s > 2\sqrt{dn}$, $S(x)$ must be the zero polynomial.
- The fact that $Q(x, p(x)) = 0$ means that $p(x)$ is a root of Q, considering Q as polynomial in y with coefficients that are polynomials in y. That is, we consider $Q(x, y)$ as an element of the polynomial ring $R[y]$, where $R = \mathbb{F}_q[x]$. We know that for any integral domain R, if $g(y) \in R[y]$ has a root $\alpha \in R$, then $y - \alpha$ divides $g(y)$ in $R[y]$. Taking $\alpha = p(x)$,we get that $y - p(x)$ divides $Q(x, y)$. Thus it will appear when we factor $Q(x, y)$. (Multivariate polynomial rings $F[x, y]$ have unique factorization, and this factorization can be done in polynomial time for most common fields, including finite fields.)

- Reed–Solomon Codes and versions of the above decoding algorithm are widely used in practice, e.g. on CDs and in satellite communications.