AM 106: Applied Algebra **Salil Vadhan**

Lecture Notes 23

December 4, 2018

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

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

- **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 $D(r, \mathrm{Enc}(m)) \leq t$, 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. 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 Reed–Solomon Codes

- **Reed-Solomon Code:** The *q-ary Reed–Solomon code of message length k and blocklength n* is a code $\mathrm{RS} : \mathbb{F}_q^k \to \mathbb{F}_q^n$ with alphabet $\Sigma = \mathbb{F}_q$. We view the message $m = (m_0, \dots, m_{k-1}) \in \mathbb{F}_q^k$ 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 $\mathrm{RS}(m) = (p_m(\alpha_1), \dots, p_m(\alpha_n))$ where $\alpha_1, \dots, \alpha_n$ 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 $t = \lfloor (n-k)/2 \rfloor$.
  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 (as you may see in section).

- The minimum distance $n - k + 1$ of a Reed–Solomon code is the best possible for codes $\mathrm{Enc} : \Sigma^k \to \Sigma^n$ over any alphabet Σ. Indeed, since there are more choices for messages $m \in \Sigma^k$ than there are choices for the first $k-1$ symbols of $\mathrm{Enc}(m)$, the Pigeonhole Principle says that there must be two distinct messages m, m' of length k such that $\mathrm{Enc}(m)$ and $\mathrm{Enc}(m')$ agree on the first $k-1$ symbols. That is, $D(\mathrm{Enc}(m), \mathrm{Enc}(m')) \leq n - (k-1)$.

- **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 $\mathrm{RS}(m) = (p_m(\alpha_1), \dots, p_m(\alpha_n))$ has distance at most t from $r = (\beta_1, \dots, \beta_n)$. This amounts to solving the following problem.

- **Noisy Polynomial Interpolation:** Given n pairs $(\alpha_1, \beta_1), \dots, (\alpha_n, \beta_n) \in \mathbb{F}_q \times \mathbb{F}_q$ with $\alpha_1, \dots, \alpha_n$ distinct, we want to find the (unique) polynomial p of degree at most $d = k-1$ such that $p(\alpha_i) = \beta_i$ for at least $n - t = \lceil (n+k)/2 \rceil$ values of i (if such a polynomial p exists).

- **Thm:** The Noisy Polynomial Interpolation problem can be solved in polynomial time.

  Proof: The first step of the algorithm is to find polynomials $W(x)$ and $E(x)$ such that:

  - $W(\alpha_i) = \beta_i \cdot E(\alpha_i)$ for $i = 1, \dots, n$.

– W has degree strictly smaller than $n - t = \lceil (n+k)/2 \rceil$.
– E has degree at most $t = \lfloor (n-k)/2 \rfloor$.
– At least one of W or E is nonzero.

We can do this by solving a system of linear equations in the field F. With the degree constraints, we have $n - t$ coefficients to choose for W and $t + 1$ coefficients to choose for E, so a total of $n+1$ unknowns. We have n conditions $W(\alpha_i) = \beta_i \cdot E(\alpha_i)$, each of which imposes a linear constraint on the coefficients of W and E. Consequently we can find a solution where not all the coefficients are zero by Gaussian elimination, which takes polynomial time in finite fields.

Once we have polynomials $W(x)$ and $E(x)$, we claim that $W(x) = p(x)E(x)$ in $\mathbb{F}_q[x]$. This is because both sides are polynomials of degree smaller than $n - t$ (note that the degree of $p(x)E(x)$ is at most $t + k - 1 \leq n - t$), yet they agree in at least $n - t$ locations, so they must be identical as polynomials. Indeed, for every i such that $p(\alpha_i) = \beta_i$, we have $W(\alpha_i) = \beta_i E(\alpha_i) = p(\alpha_i)E(\alpha_i)$. Thus, we can find $p(x)$ simply by doing long division of $W(x)$ by $E(x)$, and $p(x)$ will be the quotient (with no remainder). Note that from our guarantee that $W(x)$ or $E(x)$ is nonzero and the equation $W(x) = p(x)E(x)$, it follows that $E(x)$ is nonzero and that the division will have a zero remainder. If the algorithm ends up with the zero polynomial for $E(x)$ or with a nonzero remainder during the division, it means that no solution $p(x)$ exists. ■

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