AM 106/206: Applied Algebra	**Prof. Salil Vadhan**
Lecture Notes 22	
	December 2, 2010

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. 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 $m = (m_0, \ldots, m_{k-1}) \in \mathbb{F}_q^k$ 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 $\mathrm{RS}(m) = (p_m(\alpha_1), \ldots, p_m(\alpha_n))$ 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 $\mathrm{RS}(m) = (p_m(\alpha_1), \ldots, p_m(\alpha_n))$ 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 $\rho = 1/9$). On PS10, you will show how to improve this and decode up to $t = (n - k)/2$ errors, the same as guaranteed (inefficiently) by the minimum distance.

  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 $\sqrt{n/d}$.

You're showing how to do this on Problem Set 9.

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.