- Also, by (1): $\sum_{k=1}^{m} \omega_k = \sum_{\substack{k\in[m]\\ i\in[\ell]}} z_{i,k} \geqslant 2\delta \ell m.$

By Cauchy-Schwarz's, $\sum \omega_k^2 \geqslant (\sum \omega_k)^2/m$,
$\Rightarrow \langle \omega, \omega \rangle \geqslant (2\delta m \ell)^2/m = 4\delta^2 \ell^2 m.$

- This means, by (3), that:
$4\delta^2\ell^2 m \leqslant \langle \omega, \omega \rangle \leqslant \ell m + 2\delta^2 \ell^2 m$
$\Rightarrow \quad 2\delta^2 \ell \leqslant 1$
$\because \delta > 0 \quad \Rightarrow \quad \ell \leqslant 1/2\delta^2 \leqslant 1/2\varepsilon.$ $\square$

- Thus, there are not too many codewords $(\frac{1}{2} - \sqrt{\varepsilon})$-close to x if the distance of the code is $(\frac{1}{2} - \varepsilon)$.

Can we compute this list efficiently? & locally?

- We will see that both the answers are yes!

List Decoding RS

Theorem (Sudan '95): $\exists$ randomized poly-time algorithm that given $\{(a_i, b_i) \in \mathbb{F}^2 \mid i \in [m]\}$, returns the list of all degree $\leq d$ polynomials G s.t. $\#\{i \in [m] \mid G(a_i) = b_i\} > \sqrt{2dm}$.

[i.e. for distance $> 1 - \frac{d-1}{m}$, the list decoder handles $< 1 - \sqrt{\frac{2d}{m}}$ errors.]

Proof:

- Idea – Instead of interpolating a univariate, work with a bivariate polynomial.

1) Find a nonzero $Q(X,Y) \in \mathbb{F}[X,Y]$ s.t.

$$Q(a_i, b_i) = 0 \ , \ \forall i \in [m] \ \&$$

$(1,d)$-weighted-deg $(Q) \leqslant \sqrt{2dm} =: t$.

[^{e}:= max $\{i + dj \mid$ monomial $X^i Y^j$ is in the support of $Q\}$.]

[The maximum number of monomials that Q can have $= \sum_{0 \leq j \leq t/d} (1 + t - dj)$

$= (1+t)\cdot(1+\lfloor t/d \rfloor) - \frac{d}{2}\cdot|\frac{t}{d}|(1+\lfloor t/d \rfloor)$

$\geqslant (1+t-\frac{d}{2}\cdot\lfloor t/d \rfloor)\cdot(1+\lfloor t/d \rfloor)$

$\geqslant (1+t/2)\cdot t/d = t/d + t^2/2d$

$> m.$

Since #eqns. < #unknowns, the homogeneous linear system can be solved to get a $Q(X,Y)$ in Step-1.]

2) Factor $Q(X,Y)$ using any efficient polynomial factoring algorithm (over finite $\mathbb{F}$).

3) For factors of the form $Y-P(X)$, where $\deg P \leq d$ & $\#\{i \in [m] \mid P(a_i)=b_i\} > t$, OUTPUT $P(x)$.

[Any $\deg \leq d$ $G(X)$ that "fits" $>t$ points yields: $\begin{cases} \deg Q(X,G(X)) \leq t, \text{ \&} \\ Q(X,G(X))=0 \text{ on } >t \text{ distinct } a_i\text{'s}. \end{cases}$

$\Rightarrow Q(X,G(X))=0 \Rightarrow Y-G(X) \mid Q(X,Y).$]

$\square$

Corollary: RS, of distance $1-\frac{d-1}{m}$, has a list decoder handling $< 1-\sqrt{2d/m}$ errors & list-size $\leq \sqrt{2m/d}$.

Proof:

- In the proof above, the list-size is $\leq \deg_Y Q \leq t/d = \sqrt{2m/d}$. $\square$

Local List Decoding

Defn: Let $E:\{0,1\}^n \to \{0,1\}^m$ be an ecc & let $\varepsilon := 1/2 - \rho$ for $\rho \in (0, 1/2)$.

An algorithm $\mathcal{D}$ is a local list decoder for E handling ρ errors, if $\forall x \in \{0,1\}^n$, $\forall y \in \{0,1\}^m$ with $\Delta(E(x), y) \leq \rho$

i_0 is the advice $\rightarrow$ $\exists i_0 \in [\text{poly}(n/\varepsilon)]$ s.t.

On inputs $\langle i_0, j, \text{ oracle } y \rangle$, $\mathcal{D}$ runs for $\text{poly}(\lg m, n/\varepsilon)$-time & outputs x_j with probability $\geq 2/3$.

Local list decoding WH

Theorem 1 (Goldreich-Levin, '89): Let $WH : \{0,1\}^n \to \{0,1\}^{2^n}$ & $f : \{0,1\}^n \to \{0,1\}$ be an oracle s.t. $\exists x \in \{0,1\}^n$, $\Pr_z[f(z) = WH(x)_z] \geq \frac{1}{2} + \frac{\varepsilon}{2}$.

Let L_f be the list of such x's.

$\exists$ poly(n/ε)-time randomized algorithm to find $L_f := \{x \mid \Delta(f, WH(x)) \leq \frac{1}{2} - \frac{\varepsilon}{2}\}$.

Proof:

- Idea – Since the corruption in f is close to $1/2$, we do not get x_i in two queries. Instead we will make a single query & the other answer we will "guess".

  To reduce the error/time in finding all $x = x_1 \cdots x_n$ we will use correlated but pairwise-independent queries.

- Fix $m = \lceil 200n/\varepsilon^2 \rceil$, $k := \lfloor \lg(m+1) \rfloor$.
- Randomly pick "points" $s_1, \ldots, s_k \in \{0,1\}^n$ & "guesses" $\sigma_1, \ldots, \sigma_k \in \{0,1\}$.

[Hope: $\exists x \in L_f, \forall i \in [k], \sigma_i = x \odot s_i$.]