Class 11 Exercises

CS250/EE387, Winter 2022

1. **(We will do this exercise/recap together in class).** Recall Sudan's algorithm from the lecture notes/videos. We are trying to list-decode a RS code of dimension k over $\mathbb{F}_q$ with evaluation points $\alpha_1, \alpha_2, \ldots, \alpha_n$.

   Given a received word $y \in \mathbb{F}_q^n$:

   - **Interpolation Step:** Interpolate a nonzero polynomial $Q(X,Y) = \sum_{i=1}^{\ell} A_i(X) Y^i$ with Y-degree ℓ and X-degree n/ℓ so that $Q(\alpha_i, y_i) = 0$ for all $i = 1, \ldots, n$.
   - **Root-Finding Step:** Factor $Q(X,Y)$ and find all factors of the form $(Y - f(X))$, where $\deg(f) < k$. For each such factor, add (the codeword corresponding to) $f(X)$ to the output list.

   Choose $\ell = \sqrt{n/k}$. Reconstruct the quantitative argument from the notes/videos to prove that this algorithm is a good list-decoding algorithm. Come up with a statement like: *"the RS code is (ρ, L)-list-decodable with $L =$[something to do with the rate of the code], provided that ρ is at most [something to do with the rate of the code]."*

   Note: The notation for the algorithm above is slightly different than the notation from the videos/notes. (In particular, ℓ is playing a slightly different role). Don't just copy the notes!

   Hint: As a reminder, the outline of the argument is:

   - Argue that you can do the interpolation step.
   - Suppose that we should return $f(X)$, meaning that its encoding is within the radius ρ of y. Consider $R(X) = Q(X, f(X))$. Argue that if ρ is small enough, you can ensure that $R(X)$ has lots of roots and so has to be identically zero. How small do you need to take ρ?
   - Argue that if $R(X) \equiv 0$ then we'll return $f(X)$.
   - Make the desired statement about list-decoding.

   Solution

   Following the outline,

   - We can do the interpolation since there are n constraints and $(\ell + 1)(n/\ell + 1) > n$ coefficients in the resulting linear system.
   - The degree of $R(X)$ is at most $n/\ell + \ell \cdot k$. With the choice of $\ell = \sqrt{n/k}$, this is $\deg(R) \leq 2\sqrt{nk}$. However, $R(X)$ has a root for every location where $f(\alpha_i) = y_i$. By assumption there are at least $(1-\rho)n$ of those. So we have that $R(X) \equiv 0$ as long

as

$$(1-\rho)n > 2\sqrt{nk},$$

or

$$1-\rho > 2\sqrt{k/n} = 2\sqrt{R}$$

where R is the rate of the code. Thus, we can list-decode up to radius

$$\rho = 1 - 2\sqrt{R}.$$

- If $R(X) \equiv 0$ then $Q(X, f(X)) \equiv 0$, which means that $(Y - f(X)) | Q(X, Y)$ (just like if $P(a) = 0$ then $(X - a)|p(X)$).
- We conclude that as long as $\rho \leq 1 - 2\sqrt{R}$, that a RS code of rate R is $(\rho, \sqrt{1/R})$-list-decodable. Here, we got the list size since there are at most $\ell = \sqrt{n/k} = 1/\sqrt{R}$ roots $f(X)$ of a polynomial $Q(X, Y)$ of Y-degree ℓ.

2. In this exercise, we'll see a list-decoding algorithm (which might look somewhat familiar...) for a class of codes called *Chinese Remainder Codes* (c.f. Problem 2.1 on HW3). Below, $\mathbb{Z}_N$ refers to the integers $\{0, 1, \ldots, N-1\}$ with arithmetic mod N.

   These codes are based on the *Chinese Remainder Theorem*:

   Theorem 1. *Let $p_1, \ldots, p_t$ be relatively prime. Let $P = \prod_{i=1}^t p_i$. Fix $a_1, \ldots, a_t \in \mathbb{Z}_P$. There is a unique $m \in \mathbb{Z}_P$ so that $m \equiv a_i \mod p_i$ for all $i \in [t]$.*

   This inspires the following code[1]:

   Definition 1. *Fix $p_1 < p_2 < \cdots < p_n$ relatively prime. Let $N = \prod_{i=1}^n p_i$ and let $K = \prod_{i=1}^k p_i$. Define an encoding map $E : \mathbb{Z}_K \to \mathbb{Z}_{p_1} \times \mathbb{Z}_{p_2} \times \cdots \times \mathbb{Z}_{p_n}$ given by*

   $$E(m) = (m \mod p_1, m \mod p_2, \ldots, m \mod p_n).$$

   The Chinese Remainder Code *with parameters k and n defined by $p_1, \ldots, p_n$ is the set of codewords $\{E(m) : m \in \mathbb{Z}_K\}$.*

   In your homework (HW3, problem 2.1), you will show that these codes have distance at least $n - k + 1$, matching RS codes. But what about list-decoding?

   (a) Consider the following list-decoding algorithm. Let $y = (y_1, \ldots, y_n) \in \mathbb{Z}_{p_1} \times \cdots \times \mathbb{Z}_{p_n}$ be a received word. Our goal is to find all of the $m \in \mathbb{Z}_K$ so that $\text{dist}(E(m), y) \leq \rho n$.

   Input: $y \in \mathbb{Z}_{p_1} \times \cdots \times \mathbb{Z}_{p_n}$, parameters ℓ, F to be determined.

   - Let $r \in \mathbb{Z}_N$ be the unique element so that $r \equiv y_i \mod p_i$ for all $i \in [n]$.
   - **Interpolation Step:** Find $a = (a_0, a_1, \ldots, a_\ell)$ so that $a \neq \vec{0}$ and so that the following hold:
     - $|a_i| \leq F/K^i$ for all $i = 0, \ldots, \ell$.
     - $\sum_{i=0}^{\ell} a_i r^i \equiv 0 \mod N$.
   - **Root-finding Step:** Return the roots of $A(X) = \sum_{i=0}^{\ell} a_i X^i$. (Here, this polynomial is over the integers, not modulo anything).

   There is no question for this part, just make sure the algorithm parses.

   (b) Suppose that we can do the **Interpolation Step** with our chosen ℓ, F. Let $m \in \mathbb{Z}_K$ and suppose that $\text{dist}(E(m), y) \leq \rho n$. Show that, if ρ is not too large, then $A(m) = 0$, where $A(X) = \sum_i a_i X^i$.

   Hint: Follow the following outline:

   i. Suppose that $E(m)$ and y agree in position i. Explain why $A(m) \equiv 0 \mod p_i$.

   ii. By the previous part, if $\text{dist}(E(m), y) \leq \rho n$, then there are $(1-\rho)n$ values of i so that $A(m) \equiv 0 \mod p_i$. Use the conditions on the a_i to bound $|A(m)| \leq$ [*something*] and use the Chinese Remainder Theorem to conclude that $A(m) \equiv 0$, provided that ρ is not too big.

   How big can ρ be, in terms of ℓ, F, and the p_i's? (It will be useful later to simplify your answer to be in terms of ℓ, F and p_1, the smallest of the p_i's).

[1] Notice that the alphabet is different for each symbol, so it doesn't strictly match our definition of a code, but let's go with it.

Solution

Notice that, over $\mathbb{Z}$,

$$|A(m)| < \sum_{i=0}^{\ell} (F/K^i)K^i = (\ell+1)F,$$

using the fact that $m < K$. If we remove the absolute values, we can treat $A(m)$ as living in $\mathbb{Z}_P$ for any $P \geq 2(\ell+1)F$. If $E(m)$ and y agree in at least $(1-\rho)n$ places, then

$$A(m) \equiv 0 \mod p_i$$

for at least $(1-\rho)n$ different values of i. Let $P = \prod_{i:E(m)_i = y_i} p_i$.
By the CRT, there is a unique value $M \in \mathbb{Z}_P$ so that $M \equiv 0 \mod p_i$ for all i. One the one hand, $M = 0$ is such an M. On the other hand, if $P \geq 2(\ell+1)F$, then $A(m)$ is that unique value. So we conclude that if $P \geq 2(\ell+1)F$, then

$$A(m) = 0.$$

Therefore, our second step—returning all the roots of A—will indeed include m in the list.
We can simplify this requirement a bit by observing that it's enough for

$$p_1^{(1-\rho)n} \geq 2(\ell+1)F,$$

since p_1 is the smallest of the p's.

(c) Observe that the previous part shows that, *if* we can do the Interpolation Step, and *if* ρ is not too big, any m that satisfies $\mathrm{dist}(E(m), y) \leq \rho n$ will be returned in the root-finding step. That is, we will have a correct list-decoding algorithm, up to radius ρ!
(For this question, if you don't immediately observe this, then explain why this is the case!)

(d) Towards doing the Interpolation Step, prove the following lemma.

Lemma 2. *Fix $r \in \mathbb{Z}_N$. Suppose that $B_0, \ldots, B_\ell \in \mathbb{Z}$ are such that $B_i > 0$, and $\prod_{i=0}^{\ell} B_i > N$. Show that there exist $a_0, \ldots, a_\ell \in \mathbb{Z}$ (not all zero), so that $|a_i| < B_i$ for all i, and so that*

$$\sum_{i=0}^{\ell} a_i r^i \equiv 0 \mod N.$$

Hint: Consider the map $f : \mathbb{Z}_{B_0} \times \cdots \times \mathbb{Z}_{B_\ell} \to \mathbb{Z}_N$ given by $f(x_0, \ldots, x_\ell) = \sum_{i=0}^{\ell} x_i r^i \mod N$. Use the pigeonhole principle.

Solution

Following the hint, let f be as above. Since $\prod_{i=0}^{\ell} B_i > N$, there are some distinct $\vec{x}, \vec{x'}$ so that $f(\vec{x}) = f(\vec{x}')$. Let $a_i = x_i - x_i'$ (over $\mathbb{Z}$, not over $\mathbb{Z}_{B_i}$). Notice that

$|a_i| \leq B_i$ as required. Moreover,

$$\sum_{i=0}^{\ell} a_i r^i = \sum_{i=0}^{\ell} x_i r^i - \sum_{i=0}^{\ell} x_i' r^i \equiv 0 \mod N.$$

(e) Suppose that you don't care about the efficiency of the **Interpolation Step.** Using the previous part, what relationship do N, F, K, ℓ need to satisfy in order for you to guarantee the **Interpolation Step** can be done?

Translate this to a guarantee on p_n, n, k as well as F, ℓ.

Solution

We apply the lemma with $B_i \leftarrow F/K_i$, and we see that the lemma applies as long as

$$N < \prod_{i=0}^{\ell} F/K^i = F^{\ell+1} K^{-\ell(\ell+1)/2}.$$

Using the fact that p_n is the largest, it is enough for

$$p_n^{n+k\ell(\ell+1)/2} < F^{\ell+1}.$$

(f) Choose $\ell = \sqrt{n/k}$. Put the previous parts together (and pick an appropriate F) to produce a statement like "as long as $\rho \leq$ ____, the code is (ρ, ____)-list-decodable with the algorithm above." The ___'s should be in terms of k, n, and the p_i's. It might be convenient to get a guarantee in terms of $\kappa := \log(p_n)/\log(p_1)$.

You may also assume that $p_n \gg \ell$ and use big-Oh notation in your bound to simplify it.

Solution

From part (b), we need

$$p_1^{(1-\rho)n} \geq 2(\ell+1)F$$

and from part (d) we need

$$p_n^{n+k\ell(\ell+1)/2} < F^{\ell+1}.$$

Using the first equation, let's set

$$F = \frac{p_1^{(1-\rho)n}}{2(\ell+1)}.$$

Plugging this into the second equation and taking $\ell + 1$'st roots, we need

$$p_n^{n/(\ell+1)+k\ell/2} < \frac{p_1^{(1-\rho)n}}{2(\ell+1)}.$$

Now we take logs base p_n and get

$$\frac{n}{\ell+1} + \frac{k\ell}{2} < \frac{(1-\rho)n}{\kappa} - \log_{p_n}(2(\ell+2))$$

Since $p_n \gg \ell$, the last term is $o(1)$, and we can ignore the $+1$ in the denominator of the first term. Dividing by n, we get

$$\kappa\left(\frac{1}{\ell}+\frac{k\ell}{2n}\right) < 1-\rho$$

and plugging in $\ell \leftarrow \sqrt{n/k}$, we get

$$\kappa\left(\sqrt{k/n}+\sqrt{k/n}/2\right) < 1-\rho,$$

or

$$\rho \leq 1-\frac{3\kappa}{2}\sqrt{k/n}.$$

If ρ satisfies this, we conclude that we can do the interpolation step, and that for any m we wnat to return the root-finding step returns it. FInally, we observe that the root-finding step returns at most $\ell = \sqrt{n/k}$ things, so we get:

> *Suppose that $\rho \leq 1-\frac{3\kappa}{2}\sqrt{k/n}$. Then the CRT code is $(\rho, \sqrt{n/k})$-list-decodable.*

(g) Compare this (both the algorithm and the result) with the Sudan (or Guruswami-Sudan) algorithm for Reed-Solomon codes.

(Bonus.) Fun thing to think about, if you are familiar with polynomial quotient rings: With the CRT codes, the i'th symbol was $m \mod p_i$. One way to view an RS code is that the i'th symbol is $f(X) \mod (X-\alpha_i)$. Push this analogy as far as you can in the context of the algorithm we just developed.

Solution

The framework is very similar! We have an interpolation step and a root-finding step. The analysis proceeds by showing that a "correct" message is a root of a polynomial that we interpolate.

The quantitative result is also pretty similar. It's not clear what the "right" notion of the Johnson bound is for codes with different alphabets for each symbol (c.f. HW3, problem 2.1 for more on interpreting this), but if we just take k, n for granted, we can write $\rho \leq 1-\tilde{O}(\sqrt{k/n})$ in both cases, where the $O(\cdot)$ in the CRT case depends on κ (which is actually not a constant).

(There's lots more to say here about comparing and contrasting these two algorithms!)

(h) **(Bonus).** What if you want the **Interpolation Step** to be efficient? Would you have to change the parameters?

Solution

Check out the paper "Chinese Remaindering with Errors" by Goldreich, Ron and Sudan here: https://eccc.weizmann.ac.il/report/1998/062/revision/4/download/