Class 5 Exercises

CS250/EE387, Winter 2022

In this class, we'll investigate/develop the *Berlekamp-Massey Algorithm* for decoding Reed-Solomon codes. Some notation:

- We will be working with a RS code $C \subseteq \mathbb{F}_q^n$ with length $n = q-1$ over $\mathbb{F}_q$ and with evaluation points $\gamma, \gamma^2, \ldots, \gamma^{q-1}$.
- We will try to decode C from e errors. Suppose that $v \in \mathbb{F}_q^n$ is the received word, so $v = c + p$ for $c \in C$ and $p \in \mathbb{F}_q^n$ is an error vector so that $\mathrm{wt}(p) \leq e$.
- Let $p = (p_0, \ldots, p_{n-1})$ be the error vector, and let

$$E = \{i : p_i \neq 0\}$$

be the locations of the errors.
- Let $d = n - k + 1$ be the distance of the RS code, and assume that $e \leq \left\lfloor \frac{d-1}{2} \right\rfloor$, so that unique decoding is possible.

Now onto the questions.

1. The first step of the Berlekamp-Massey algorithm is to compute the *syndrome* $s = Hv$, where H is the parity-check matrix for C. Write $s = (s_1, \ldots, s_{d-1})$, and let $s(Z) = \sum_{i=1}^{d-1} s_i Z^i$.

   Show that
   $$s(Z) = \sum_{i \in E} p_i \left(\frac{\gamma^i Z - (\gamma^i Z)^d}{1 - \gamma^i Z} \right).$$

   Hint. Use the structure of the parity-check matrix H, and first show that $s(Z) = \sum_{i \in E} p_i \sum_{\ell=1}^{d} (\gamma^i Z)^\ell$.

2. Let $\sigma(Z) = \prod_{i \in E}(1 - \gamma^i Z)$. Explain why we will be in good shape for decoding if we can figure out what σ is.

3. Consider $\sigma(Z) \cdot s(Z)$. Show that this can be written as
   $$\sigma(Z) \cdot s(Z) = w(Z) + Z^d r(Z),$$
   where $w(Z)$ and $r(Z)$ are polynomials, and $\deg(w) \leq e$. Write down an expression for $w(Z)$.

   At this point please fill out the number poll to "3".

4. Show that the following are true:

   (a) For all $r \in E$, $w(\gamma^{-r}) = p_r \cdot \prod_{j \in E \setminus \{i\}} (1 - \gamma^{j-i})$.

   At this point please fill out the number poll to "4(a)".

(b) **(Optional:)** $w(Z)$ and $\sigma(Z)$ are relatively prime. That is, there is no polynomial $g(Z)$ (other than $g(Z) \equiv 1$) that divides them both.

(Note: if you don't feel like showing this, take it as given and skip this part; but we will probably come back together as a class after problem 4, so if you get this far you may as well think about it!).

At this point please fill out the number poll to "4(b)".

5. The previous part implies that, for all r with $e + 1 \leq r \leq d - 1$, we have

$$\text{coefficient on } Z^r \text{ in } s(Z)\sigma(Z) = 0.$$

What is that coefficient, in terms of the coefficients of s (which we know) and the coefficients of σ (which we don't know)? Write down a system of $d - e - 1$ linear constraints that the coefficients of σ must satisfy. Your constraints should be in terms of the s_i. Explain why there is at least one solution to this system of equations.

At this point please fill out the number poll to "5".

6. Suppose we were to solve your system of equations to obtain $(\tilde{\sigma}_0, \ldots, \tilde{\sigma}_{n-1})$ and a corresponding polynomial $\tilde{\sigma}(Z) = \sum_{i=0}^{e} \tilde{\sigma}_i Z^i$. Explain why we can write

$$s(Z)\tilde{\sigma}(Z) = \tilde{w}(Z) + Z^d \tilde{r}(Z)$$

for some polynomials $\tilde{w}(Z), \tilde{r}(Z)$ with $\deg(\tilde{w}) \leq e$.

At this point please fill out the number poll to "6".

7. Show that $\tilde{\sigma}(Z)w(Z) = \sigma(Z)\tilde{w}(Z)$.

<u>Hint:</u> Consider $s(Z)\sigma(Z)\tilde{\sigma}(Z)$.

8. Explain how, given $\tilde{\sigma}$ and $\tilde{w}$, to find σ and w.

<u>Hint</u>: The answer to part of it is that $\sigma(Z) = \tilde{\sigma}(Z)/gcd(\tilde{\sigma}, \tilde{w})$...but why?

<u>Hint</u>: Question 7, along with Part 4(b), might be useful.

At this point please fill out the number poll to "8".

9. Put all the pieces together to write down an efficient (polynomial-time) algorithm to recover the codeword c given v. What is the running time of your algorithm, in terms of the number of operations over $\mathbb{F}_q$? Do you see ways you might be able to speed it up?

If it helps, finding the gcd of two degree-D polynomials (with, say, Euclid's algorithm) takes $O(D^2)$ operations over $\mathbb{F}_q$. Finding the roots of a degree-D polynomial in $\mathbb{F}_q$ can be done with $O(D^2 \log q)$ operations over $\mathbb{F}_q$.