Berlekamp–Massey algorithm

The **Berlekamp–Massey algorithm** is an algorithm that will find the shortest linear-feedback shift register (LFSR) for a given binary output sequence. The algorithm will also find the minimal polynomial of a linearly recurrent sequence in an arbitrary field. The field requirement means that the Berlekamp–Massey algorithm requires all non-zero elements to have a multiplicative inverse.[1] Reeds and Sloane offer an extension to handle a ring.[2]

Elwyn Berlekamp invented an algorithm for decoding Bose–Chaudhuri–Hocquenghem (BCH) codes.[3][4] James Massey recognized its application to linear feedback shift registers and simplified the algorithm.[5][6] Massey termed the algorithm the LFSR Synthesis Algorithm (Berlekamp Iterative Algorithm),[7] but it is now known as the Berlekamp–Massey algorithm.

Description of algorithm

The Berlekamp–Massey algorithm is an alternative to the Reed–Solomon Peterson decoder for solving the set of linear equations. It can be summarized as finding the coefficients Λ_j of a polynomial $\Lambda(x)$ so that for all positions i in an input stream S:

$$S_{i+\nu} + \Lambda_1 S_{i+\nu-1} + \cdots + \Lambda_{\nu-1} S_{i+1} + \Lambda_\nu S_i = 0.$$

In the code examples below, $C(x)$ is a potential instance of $\Lambda(x)$. The error locator polynomial $C(x)$ for L errors is defined as:

$$C(x) = C_L x^L + C_{L-1} x^{L-1} + \cdots + C_2 x^2 + C_1 x + 1$$

or reversed:

$$C(x) = 1 + C_1 x + C_2 x^2 + \cdots + C_{L-1} x^{L-1} + C_L x^L.$$

The goal of the algorithm is to determine the minimal degree L and $C(x)$ which results in all syndromes

$$S_n + C_1 S_{n-1} + \cdots + C_L S_{n-L}$$

being equal to 0:

$$S_n + C_1 S_{n-1} + \cdots + C_L S_{n-L} = 0, \qquad L \le n \le N-1.$$

Algorithm: $C(x)$ is initialized to 1, L is the current number of assumed errors, and initialized to zero. N is the total number of syndromes. n is used as the main iterator and to index the syndromes from 0 to $N-1$. $B(x)$ is a copy of the last $C(x)$ since L was updated and initialized to 1. b is a copy of the last discrepancy d (explained below) since L was updated and initialized to 1. m is the number of iterations since L, $B(x)$, and b were updated and initialized to 1.

Each iteration of the algorithm calculates a discrepancy d. At iteration k this would be:

$$d \leftarrow S_k + C_1 S_{k-1} + \cdots + C_L S_{k-L}.$$

If d is zero, the algorithm assumes that $C(x)$ and L are correct for the moment, increments m, and continues.

If d is not zero, the algorithm adjusts $C(x)$ so that a recalculation of d would be zero:

$$C(x) \leftarrow C(x) - (d/b) x^m B(x).$$

The x^m term *shifts* B(x) so it follows the syndromes corresponding to b. If the previous update of L occurred on iteration j, then $m = k - j$, and a recalculated discrepancy would be:

$$d \leftarrow S_k + C_1 S_{k-1} + \cdots - (d/b)(S_j + B_1 S_{j-1} + \cdots).$$

This would change a recalculated discrepancy to:

$$d = d - (d/b)b = d - d = 0.$$

The algorithm also needs to increase L (number of errors) as needed. If L equals the actual number of errors, then during the iteration process, the discrepancies will become zero before n becomes greater than or equal to $2L$. Otherwise L is updated and algorithm will update $B(x)$, b, increase L, and reset $m = 1$. The formula $L = (n + 1 - L)$ limits L to the number of available syndromes used to calculate discrepancies, and also handles the case where *L* increases by more than 1.

Code sample

The algorithm from Massey (1969, p. 124) for an arbitrary field:

```
  polynomial(field K) s(x) = ... /* coeffs are s_j; output
sequence as N-1 degree polynomial) */
  /* connection polynomial */
  polynomial(field K) C(x) = 1;  /* coeffs are c_j */
  polynomial(field K) B(x) = 1;
  int L = 0;
  int m = 1;
  field K b = 1;
  int n;

  /* steps 2. and 6. */
  for (n = 0; n < N; n++) {
      /* step 2. calculate discrepancy */
      field K d = s_n + \Sigma_{i=1}^L c_i * s_{n-i};

      if (d == 0) {
          /* step 3. discrepancy is zero; annihilation continues
*/
          m = m + 1;
      } else if (2 * L <= n) {
          /* step 5. */
          /* temporary copy of C(x) */
          polynomial(field K) T(x) = C(x);

          C(x) = C(x) - d b^{-1} x^m B(x);
          L = n + 1 - L;
          B(x) = T(x);
          b = d;
          m = 1;
```

```
        } else {
            /* step 4. */
            C(x) = C(x) - d b^{-1} x^m B(x);
            m = m + 1;
        }
    }
    return L;
```

In the case of binary GF(2) BCH code, the discrepancy d will be zero on all odd steps, so a check can be added to avoid calculating it.

```
/* ... */
  for (n = 0; n < N; n++) {
      /* if odd step number, discrepancy == 0, no need to
calculate it */
      if ((n&1) != 0) {
          m = m + 1;
          continue;
      }
/* ... */
```

See also

- Reed–Solomon error correction
- Reeds–Sloane algorithm, an extension for sequences over integers mod *n*
- Nonlinear-feedback shift register (NLFSR)

References

1. *Reeds & Sloane 1985, p. 2*
2. *Reeds, J. A.; Sloane, N. J. A. (1985), "Shift-Register Synthesis (Modulo n)" (http://neilsloane.com/doc/Me111.pdf) (PDF), SIAM Journal on Computing, **14** (3): 505–513, CiteSeerX 10.1.1.48.4652 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.48.4652), doi:10.1137/0214038 (https://doi.org/10.1137%2F0214038)*

3. Berlekamp, Elwyn R. (1967), Nonbinary BCH decoding, International Symposium on Information Theory, San Remo, Italy

4. Berlekamp, Elwyn R. (1984) [1968], Algebraic Coding Theory (Revised ed.), Laguna Hills, CA: Aegean Park Press, ISBN 978-0-89412-063-3. Previous publisher McGraw-Hill, New York, NY.

5. Massey, J. L. (January 1969), "Shift-register synthesis and BCH decoding" (http://crypto.stanford.edu/~mironov/cs359/massey.pdf) (PDF), IEEE Transactions on Information Theory, IT-15 (1): 122–127, doi:10.1109/TIT.1969.1054260 (https://doi.org/10.1109%2FTIT.1969.1054260)

6. Ben Atti, Nadia; Diaz-Toca, Gema M.; Lombardi, Henri (April 2006), "The Berlekamp–Massey Algorithm revisited" (http://hlombardi.free.fr/publis/ABMAvar.html), Applicable Algebra in Engineering, Communication and Computing, **17** (1): 75–82, CiteSeerX 10.1.1.96.2743 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.96.2743), doi:10.1007/s00200-005-0190-z (https://doi.org/10.1007%2Fs00200-005-0190-z), S2CID 14944277 (https://api.semanticscholar.org/CorpusID:14944277)

7. Massey 1969, p. 124

External links

- "Berlekamp-Massey algorithm" (https://www.encyclopediaofmath.org/index.php?title=Berlekamp-Massey_algorithm), *Encyclopedia of Mathematics*, EMS Press, 2001 [1994]
- Berlekamp–Massey algorithm (https://web.archive.org/web/20120716181541/http://planetmath.org/encyclopedia/BerlekampMasseyAlgorithm.html) at PlanetMath.
- Weisstein, Eric W. "Berlekamp–Massey Algorithm" (https://mathworld.wolfram.com/Berlekamp-MasseyAlgorithm.html). *MathWorld*.
- GF(2) implementation in Mathematica (https://code.google.com/p/lfsr/)
- (in German) Applet Berlekamp–Massey algorithm (http://www.informationsuebertragung.ch/indexAlgorithmen.html)
- Online GF(2) Berlekamp-Massey calculator (https://berlekamp-massey-algorithm.appspot.com/)