Linear Algebra Project : Hamming Code

Seth Reed, Alexandra Darling, Shirley Shi, Tyler West

What the Code Does

- Richard Hamming 1950
- Linear error detecting code
- Computes hamming distance
- Hamming distance is the number of positions at which the corresponding symbols/numbers are different

Hamming Code

```
function Hamming() {

 function shortestStrand(strand1, strand2){
   if (strand1.length > strand2.length){
     return strand2;
   } else {
     return strand1
   };
 };

 function strandCounting(shortest, splitStrand1, splitStrand2){
   var count = 0
   for (var i = 0; i < shortest.length; i++) {
     if (splitStrand1[i] != splitStrand2[i]) {
       count+=1
     };
 };
   return count;
 };

 this.compute = function(strand1, strand2) {
   var splitStrand1 = strand1.split("");
   var splitStrand2 = strand2.split("");
   var shortest     = shortestStrand(splitStrand1,splitStrand2)
   var result       = strandCounting(shortest, splitStrand1,
splitStrand2);
   return result;
 };
};

exports.Hamming = Hamming;
```

Hamming Test Code

```
var Hamming = require('./hamming').Hamming;

describe('Hamming', function(){
 var hamming = new Hamming();
it('no difference between empty strands', function() {
   var result = hamming.compute("","");
   expect(result).toBe(0);
 });
it('no difference between identical strands', function() {
   var result = hamming.compute("110001", "110001");
   expect(result).toBe(0);
 });
it('hamming distance in small strands', function() {
   var result = hamming.compute("101", "010");
   expect(result).toBe(3);
 });
it('hamming distance in off by one strand', function() {
   var result = hamming.compute("10010010001", "10010010011")
   expect(result).toBe(9);
 });
it('ignores extra length on other strand when longer', function() {
   var result = hamming.compute("110101", "11000101010101")
   expect(result).toBe(3);
 });
 it('ignores extra length on original strand when longer', function() {
   var result = hamming.compute("10101010101011111",
"10101101010101010101010101")
   expect(result).toBe(5);
 });
});
```

Use of Linear Algebra

Hamming (7,4) : Linear error-correcting code that encodes four bits of data into seven bits by adding three parity bits.

Parity Bit: bit added to the end of a string of binary code that indicates whether the number of bits in the string with the value one is even or odd.

Hamming Codes can be computed in linear algebra terms through matrices.

Hamming Matrices

Code Generator Matrix G :

Parity-Check Matrix H:

$$\mathbf{G} := \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad \mathbf{H} := \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Example

Suppose we want to transmit 1011 over a noisy communications channel. Take the product of G and (1011) , with entries modulo 2, to determine transmitted codeword x:

$$\mathbf{x} = \mathbf{Gp} = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \\ 1 \\ 2 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

The receiver can check if the received codeword is correct. If the product of H and x is equal to zero, the receiver can conclude that no error has occurred.

If not, then the product gives us the position of the bit with the wrong parity.

Decoding

Received data needs to be decoded back into original four bits.

$$\mathbf{R} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Then the product of R and vector (0, 1, 1, 0, 0, 1, 1) gives us

$$\mathbf{p_r} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

Resources

- https://github.com/JonahMoses/exercism/tree/master/javascript/hamming
- https://en.wikipedia.org/wiki/Hamming_code