Crime investigation: an introduction to error-correcting codes

Gerry Myerson*

Abstract

An expository introduction to coding theory, inspired by a problem from Norman Do's Puzzle Corner.

The puzzle

> While questioning a witness, a judge is only allowed to ask questions which are to be answered 'yes' or 'no'. The judge has carefully calculated that, as long as the witness answers every question truthfully, then she can solve the case in no more than 91 questions. Show that the judge can solve the case in no more than 105 questions if it is known that the witness may answer at most one question falsely.

In fact, under the stated conditions, the judge can solve the case in 98 questions. It isn't necessary for her, or the witness, to understand anything about coding theory to implement the solution, but the solution is best understood in the context of error-correcting codes.

Error-correcting codes

Coding theory is concerned with the efficient transmission of messages across space and/or time in such a way that errors in the transmission can be detected and/or corrected. For our purposes, a 'message' will be a finite string of 0s and 1s. A code will be a collection of such messages, or codewords, all of the same length. If two codewords differ in a single bit — if, say, $a = 011010111100$ and $b = 011010110100$ are both codewords — then it will be impossible for anyone receiving the message a to know whether the intended message was a, or whether the intended message was b, and a single error occurred in transmission.

Perhaps the simplest example of a one-error-detecting code is $\{\,00, 11\,\}$. If one error is made in transmitting either codeword, the result will be 01 or 10, neither of which is a codeword. Thus, the recipient will be able to tell that an error has

Received 28 April 2008; accepted for publication 16 December 2008.
*Mathematics, Macquarie University, NSW 2109. E-mail: gerry@maths.mq.edu.au

Dedicated to the memory of the late Peter Pleasants, former colleague at Macquarie and enthusiastic contributor to the Puzzle Corner.

occurred. Of course, the recipient will be unable to tell *what* error occurred; on receiving 01, the recipient will be unable to tell whether the intended message was 00 or 11.

Perhaps the simplest example of a single-error-correcting code is $\{\,000, 111\,\}$. Now if a single error occurs during transmission, the recipient will not only be able to detect it but to unambiguously correct it.

Crime investigation

To see what this has to do with crime investigation (or, at any rate, with puzzles about crime investigation), imagine that the judge can solve the case by asking a truthful witness a single yes–no question. Then to solve the case with a witness who may lie at most once, all she needs to do is ask the same question three times. Coding 'yes' as 1 and 'no' as 0, if there are no lies then the answers must be 000 or 111. A single lie corresponds to a single error in transmission, which can be detected and corrected.

The Hamming code

A somewhat more sophisticated one-error-correcting code is the Hamming code of length 7. This consists of the following 16 codewords:

1000011	0100101	0010110	0001111
0111100	1011010	1101001	1110000
0110011	1010101	1100110	1111111
1001100	0101010	0011001	0000000

To prove that this code can correct any single error, we need to show that every pair of codewords differs in at least three places. However, this code has some additional structure which makes it unnecessary to compare each pair of words; it is a vector space.

The set of all n-bit strings of 0s and 1s can be viewed as an n-dimensional vector space — not a real or complex vector space, but a vector space over $\boldsymbol{Z}_2 = \{\,0, 1\,\}$, the field of two elements. In this field, $1 + 1 = 0$. An n-bit string can be identified with an n-component vector with entries from $\boldsymbol{Z}_2$. Vectors are added, and multiplied by scalars, in the usual component-wise way (but the only scalars are 0 and 1).

The Hamming code given above is a four-dimensional subspace of the seven-dimensional vector space of all seven-bit strings from $\boldsymbol{Z}_2$. It is easily checked that a basis is given by

$$\{\,1000011, 0100101, 0010110, 0001111\,\}$$

The difference of any two codewords is again a codeword, and the number of bits in which any two codewords differ is just the number of ones in their difference. Since there is no non-zero codeword with fewer than three 1s, there are no two codewords that differ in fewer than three locations. Thus, the code can correct any single error in transmission.

The Hamming code has another useful property; every seven-bit string is either in the code, or differs in exactly one location from a string that is in the code. A simple counting argument establishes this, for each word in the code, together with the strings that differ from it in exactly one location, form a set of size 8, and the 'three locations' condition ensures that these sets of 8 are disjoint. There are 16 disjoint sets of 8, which accounts neatly for all 128 binary strings of length 7. For this reason the Hamming code is said to be a *perfect* code.

Crime investigation, revisited

Now a judge who can solve a case by asking a truthful witness four questions can solve the case by asking a witness seven questions, if the witness may lie at most once. The judge asks the four questions, then asks the three supplementary questions: 'If you had answered questions 2, 3 and 4 truthfully, would you have said 'yes' to an odd number of them?'; 'If you had answered questions 1, 3, and 4 truthfully, would you have said 'yes' to an odd number of them?'; 'If you had answered questions 1, 2, and 4 truthfully, would you have said 'yes' to an odd number of them?'

The judge views the seven answers received as a 7-bit string $\boldsymbol{y} = a_1a_2a_3a_4a_5a_6a_7$; remember, each 'yes' is a 1, each 'no' is a 0. If all seven answers are truthful, then $a_5 = a_2 + a_3 + a_4$, $a_6 = a_1 + a_3 + a_4$, and $a_7 = a_1 + a_2 + a_4$. These conditions are satisfied by the basis vectors given earlier for the Hamming code, hence, they are satisfied by all the codewords. The conditions are easily seen to determine a four-dimensional space, so in fact the seven answers are all truthful if and only if the string is a codeword.

Thus, if the string is a codeword, then all the answers are truthful, and the judge can solve the case. If the string is not a codeword, then there is a unique codeword from which it differs in a single location; the judge 'corrects' the string to that codeword, and solves the case.

Identifiers, matrices, and simplifications

We can make things a little easier for the judge and the witness. First, we can assign the first four questions the identifiers 011, 101, 110 and 111, in that order. Then the supplementary questions become: 'if you had answered the questions whose identifier had first bit 1 truthfully, would you have said 'yes' to an odd number of them?'; 'if you had answered the questions whose identifier had second bit 1 truthfully, would you have said 'yes' to an odd number of them?'; 'if you had answered the questions whose identifier had third bit 1 truthfully, would you have said 'yes' to an odd number of them?'

Second, we can spare the judge the work of hunting through the 16 codewords to find the closest match. The judge simply has to compute $\boldsymbol{s} = H\boldsymbol{y}$, where

$$H = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

and all arithmetic is done in $\boldsymbol{Z}_2$. Note that $H = [B \mid I]$ where B is the matrix whose columns are the identifiers given above for the first four questions and I is the 3×3 identity matrix. If we let $G = [I \mid B^t]$, where now I is the 4×4 identity matrix, so

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

then the row space of G is the nullspace of H. But the row space of G is the Hamming code, for which reason G is called the *generator matrix* for the code. Thus, $\boldsymbol{s} = \boldsymbol{0}$ if and only if $\boldsymbol{y}$ is a codeword, for which reason H is called the *parity-check matrix*. Moreover, if $\boldsymbol{s} \neq \boldsymbol{0}$, then $\boldsymbol{s}$ is one of the columns of H (since every non-zero three-bit string is a column of H), say, the jth column; then a_j is the bit of $\boldsymbol{y}$ that needs to be corrected.

For example, if $\boldsymbol{y} = 1011011$, then $H\boldsymbol{y} = 001$, which is the seventh column of H, so the seventh bit of $\boldsymbol{y}$ must be corrected, and the truthful answers are 1011010.

We can make life even easier for judge and witness. The supplementary questions can be: 'Did you lie about a question whose identifier has first bit 1?'; 'Did you lie about a question whose identifier has second bit 1?'; 'Did you lie about a question whose identifier has third bit 1?'

If the witness answered the first four questions truthfully, then he can answer 'yes' to at most one of the supplementary questions. If one of his first four answers was a lie, then he must answer all the supplementary questions truthfully, which means he must answer 'yes' to at least two of them. Thus, if he answers 'yes' to at most one of the supplementaries, he must have answered the first four truthfully, and the judge can solve the case; if he answers 'yes' to two or more of the supplementaries, then these answers must all be truthful, and the judge can easily work out which one of the first four questions elicited a deceitful answer, and solve the case. For example, if the witness responds in the affirmative to the first and third supplementaries, then he must have lied about the question with identifier 101, which was the second question.

There is a very simple relation between this scheme and the previous one. If we view the answers to this set of seven questions as the seven-bit string $\boldsymbol{z} = a_1a_2a_3a_4c_5c_6c_7$, then $c_5 = a_2 + a_3 + a_4 - a_5$, $c_6 = a_1 + a_3 + a_4 - a_6$, and $c_7 = a_1 + a_2 + a_4 - a_7$, as one can easily check by running through the logic. Thus, the new supplementary questions are giving us the same information as the old ones did, but this is a more readily usable form.

Mamma's little baby loves shortening, shortening

So much for the judge who only needs correct answers to four questions. Before we scale things up to 91 questions, we introduce the concept of *shortening* a code. To shorten a code, we choose any component, delete every codeword having a one in that component, then delete that component from the remaining codewords. For

example, to shorten the Hamming code at the fifth component, we delete the eight codewords that have a 1 in the fifth location, and delete the fifth location from the remaining eight codewords, leaving 000000, 001101, 010110, 011011, 100011, 101110, 110101 and 111000. A shortened code has length one less than the original code, and it's easy to see that if the original code had no two words differing in fewer than d locations then the same is true of the shortened code; in particular, if the original was a single-error-correcting code, then so is the shortened code. Also, if the original code was a vector space, then the shortened code is, in general, a vector space of dimension one less. Finally, what you do once, you can do again; a code can be shortened repeatedly, and the result will always be an error-correcting code if the original was.

More Hamming codes, and solving the puzzle

Now what I have been calling the Hamming code is more accurately the simplest in an infinite family of (binary) Hamming codes. Given any k, one can construct the matrix $H = [B \mid I]$ whose columns are the non-zero k-bit strings, and the matrix $G = [I \mid B^t]$ with $2^k - k - 1$ rows and $2^k - 1$ columns. Then the row-space of G is a Hamming code. It is a $2^k - k - 1$ dimensional vector space whose members are strings of length $2^k - 1$, and it has no non-zero element with fewer than three 1s, so it can correct any single error. Also, H is a parity-check matrix for this code, as the code is the nullspace of H. Decoding, error-detecting, and error-correcting can be performed for these Hamming codes in much the same way as for the length 7 code we have been working with.

Taking $k = 7$, there is a 120-dimensional error-correcting code of length 127. Shortening this code 29 times, we arrive at a 91-dimensional code of length 98, which is exactly what our judge needs in order to solve the case presented in the opening paragraphs of this essay. There is no need for judge or witness to construct the matrices G and H associated with this code. The judge simply does the following:

1. Attach to each of the 91 questions a different seven-bit identifier, each identifier having at least two 1s. Note that there are 128 seven-bit strings, of which only eight have fewer than two 1s, so there are more than enough identifiers available.
2. Ask the 91 questions.
3. Ask the seven supplementary questions: 'Did you lie about a question whose identifier has first bit 1?'; 'Did you lie about a question whose identifier has second bit 1?'; ... 'Did you lie about a question whose identifier has seventh bit 1?'

If the witness answers the 91 questions truthfully, then he can affirm at most one of the supplementary questions. If the witness fibs about one of the 91 questions, then he must answer the supplementary questions truthfully, so must affirm at least two of them. Thus, if the witness denies all, or all but one, of the supplementary questions, he must have told the truth in answer to all of the 91 questions. If the witness affirms two or more of the supplementaries, he must be answering all

the supplementaries truthfully, and the judge can then easily work out which one of the 91 questions brought forth a lie. Either way, she can then solve the case.

Judge not, lest ye be judged

Is our solution best possible? Can a cleverer judge come up with a scheme guaranteed to solve the case with only 97 questions?

Let's look at something smaller. In some other courtroom, the only thing the judge needs to know to solve a case is the remainder when the witness' age is divided by 20. There are, of course, 20 possibilities, and the judge, we assume, is unable to rule any of them out a priori. If the witness is truthful, then it is easy to design a scheme whereby the judge can solve the case with no more than five questions and might, depending on the witness' age, be able to solve it in fewer than five questions; no scheme can guarantee solving the case with fewer than five questions.

Now suppose the witness can lie at most once. Taking $k = 4$ in the previous section, we see there is an 11-dimensional Hamming code of length 15. Shortening this code six times, we arrive at a five-dimensional error-correcting code of length 9. The judge can use this to solve the case with nine yes-no questions.

But in fact the judge can solve the case with eight questions. Consider the code of length 8 which contains 11010000, 11100100, 10101010, 11111111, 00000000, and all cyclic shifts of these strings (e.g. the cyclic shifts of 11010000 are 10100001, 01000011, and so on). The reader can verify that there are 20 codewords, and every pair of codewords differs in at least three places (we owe this example to [1, Exercise 2.16]). Now all the judge has to do is assign the 20 possibilities for the witness' age to the 20 codewords (any one-one correspondence will do) and ask the witness the eight questions, 'Does the remainder when your age is divided by 20 correspond to a codeword whose first bit is a one?'; 'Does the remainder correspond to a codeword whose second bit is a one?'; ... 'Does the remainder correspond to a codeword whose last bit is a one?' The answers will either yield a codeword, or a string that differs in exactly one place from exactly one of the codewords, and in either situation the judge can solve the case.

To sum up, we can distinguish among 20 possibilities with eight questions if one lie is permitted; equivalently, there is an eight-dimensional error-correcting code with 20 codewords.

Now let's go back to the first courtroom. How many possibilities is the judge trying to distinguish? Certainly no more than 2^{91}, as she can guarantee to solve the case with 91 questions to a truthful witness. Presumably more than 2^{90}, as otherwise the 91 questions to a truthful witness could have been reduced to a smaller number. More than this we cannot say; we know only that there are m possibilities, for some m, $2^{90} < m \leq 2^{91}$.

The ability to solve the case with 97 questions is equivalent to the existence of an error-correcting code of length 97 with m codewords. If $m = 2^{91}$, then there certainly is no such code; a simple counting argument (known in the technical

literature as 'the sphere-packing bound') shows this. It is not known for which values of m there is an error-correcting code of length 97 with m codewords; indeed, it is not even known for which values of m there is an error-correcting code of length 16 with m codewords. For these reasons, we are unable to say whether the judge can solve the case with 97 questions.

Extras

That concludes our crime investigation, but only scratches the surface of error-correcting codes. Here are a few topics I feel I ought to mention, even though I can hardly say anything about them here.

The Hamming codes can correct any single error but are useless if more than one error occurs in transmission. A great deal of effort has gone into finding codes capable of correcting multiple errors by dint of having every pair of codewords differ in many locations.

In some applications it is common for errors to occur in bursts, that is, for several consecutive bits to be affected by noise in transmission. In other applications, transposition errors are common. Codes can be specially designed to deal with burst and/or transposition errors.

The triple repetition code introduced above as the simplest error-correcting code has a rate of 1/3; that is, it conveys 1 bit of information for every 3 bits transmitted. The Hamming code of length 7 has a rate of 4/7. All other things being equal, the higher the rate, the better the code. However, all other things are not equal — there are trade-offs (inequalities) connecting the rate, the length, and the error-correcting capabilities of codes. A great deal of research goes into finding the theoretical restrictions on these quantities, and into constructing codes which push up against the theoretical restrictions.

Now suppose you have a code with a high rate and a high error-correcting capability — but suppose the only way to decode an incoming message is to compare it with each word in the code to see which one is closest. That won't be very practical, if the code is large. Ease of decoding is another code characteristic that must be taken into account in applications.

All the codes discussed in this essay are binary. This makes our computers happy, but codes in which the symbols used come from a set of size exceeding two are important in both theory and practice. An example is the *International Standard Book Number* (ISBN) with which books are issued nowadays. Since 2007, the ISBNs have formed a code of length 13 on the ten digits; previously, they formed a code of length 10 on the 11 symbols 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, X.

There are many texts one can consult to follow up on some or all of these issues. Some published in the last 20 years are:

Adámek, J. (1991). *Foundations of Coding*. John Wiley.

Bierbrauer, J. (2004). *Introduction to Coding Theory*. Chapman and Hall.

Hoffman, D.G., Leonard, D.A., Lindner, C.C., Phelps, K.T., Rodger, C.A. and Wall, J.R. (1991). *Coding Theory*. Marcel Dekker.

Ling, S. and Xing, C. (2004). *Coding Theory*. Cambridge University Press.

McEliece, R.J. (2004). *The Theory of Information and Coding*, Student edn. Cambridge University Press.
Pless, V. (1998). *Introduction to the Theory of Error-Correcting Codes* John Wiley.
Pretzel, O. (1992). *Error-Correcting Codes and Finite Fields*. Oxford University Press.
Roman, S. (1997). *Introduction to Coding and Information Theory*. Springer.
Roth, R. (2006). *Introduction to Coding Theory*. Cambridge University Press.
Vermani, L.R. (1996). *Elements of Algebraic Coding Theory*. Chapman and Hall.
Welsh, D. (1998). *Codes and Cryptography*. Oxford University Press.

History

The puzzle appeared in Norman Do's Puzzle Corner 4 in this *Gazette*, 35 (2007) 202–207. It also appears at http://www.kalva.demon.co.uk/soviet/sov91.html according to which it was part of the 25th All Soviet Union Math Competition, held in 1991. This competition, with solutions, can be found in Arkadii Slinko's book, USSR Mathematical Olympiads 1989–1992, published by the Australian Mathematics Trust.

Very closely related is a problem Ulam posed in the last chapter of his autobiography, Adventures of a Mathematician (Scribner, 1976). See the solution given in [2].

References

[1] Hill, R. (1990). *A First Course in Coding Theory*. Oxford University Press.
[2] Pelc, A. (1987). Solution of Ulam's problem on searching with a lie. *J. Combin. Theory Ser. A* **44,** 129–140.