WIKIPEDIA

Block code

In coding theory, **block codes** are a large and important family of error-correcting codes that encode data in blocks. There is a vast number of examples for block codes, many of which have a wide range of practical applications. The abstract definition of block codes is conceptually useful because it allows coding theorists, mathematicians, and computer scientists to study the limitations of *all* block codes in a unified way. Such limitations often take the form of *bounds* that relate different parameters of the block code to each other, such as its rate and its ability to detect and correct errors.

Examples of block codes are Reed–Solomon codes, Hamming codes, Hadamard codes, Expander codes, Golay codes, and Reed–Muller codes. These examples also belong to the class of linear codes, and hence they are called **linear block codes**. More particularly, these codes are known as algebraic block codes, or cyclic block codes, because they can be generated using boolean polynomials.

Algebraic block codes are typically hard-decoded using algebraic decoders.

The term *block code* may also refer to any error-correcting code that acts on a block of k bits of input data to produce n bits of output data (n, k). Consequently, the block coder is a *memoryless* device. Under this definition codes such as turbo codes, terminated convolutional codes and other iteratively decodable codes (turbo-like codes) would also be considered block codes. A non-terminated convolutional encoder would be an example of a non-block (unframed) code, which has *memory* and is instead classified as a *tree code*.

This article deals with "algebraic block codes".

Contents

The block code and its parameters

Error-correcting codes are used to reliably transmit digital data over unreliable communication channels subject to channel noise. When a sender wants to transmit a possibly very long data stream using a block code, the sender breaks the stream up into pieces of some fixed size. Each such piece is called *message* and the procedure given by the block code encodes each message individually into a codeword, also called a *block* in the context of block codes. The sender then transmits all blocks to the receiver, who can in turn use some decoding mechanism to (hopefully) recover the original messages from the possibly corrupted received blocks. The performance and success of the overall transmission depends on the parameters of the channel and the block code.

Formally, a block code is an injective mapping

$$C : \Sigma^k \to \Sigma^n.$$

Here, Σ is a finite and nonempty set and k and n are integers. The meaning and significance of these three parameters and other parameters related to the code are described below.

The alphabet Σ

The data stream to be encoded is modeled as a string over some **alphabet** Σ. The size $|\Sigma|$ of the alphabet is often written as q. If $q = 2$, then the block code is called a *binary* block code. In many applications it is useful to consider q to be a prime power, and to identify Σ with the finite field $\mathbb{F}_q$.

The message length *k*

Messages are elements m of Σ^k, that is, strings of length k. Hence the number k is called the **message length** or **dimension** of a block code.

The block length *n*

The **block length** n of a block code is the number of symbols in a block. Hence, the elements c of Σ^n are strings of length n and correspond to blocks that may be received by the receiver. Hence they are also called received words. If $c = C(m)$ for some message m, then c is called the codeword of m.

The rate *R*

The **rate** of a block code is defined as the ratio between its message length and its block length:

$$R = k/n.$$

A large rate means that the amount of actual message per transmitted block is high. In this sense, the rate measures the transmission speed and the quantity $1 - R$ measures the overhead that occurs due to the encoding with the block code. It is a simple information theoretical fact that the rate cannot exceed 1 since data cannot in general be losslessly compressed. Formally, this follows from the fact that the code C is an injective map.

The distance d

The **distance** or **minimum distance** d of a block code is the minimum number of positions in which any two distinct codewords differ, and the **relative distance** δ is the fraction d/n. Formally, for received words $c_1, c_2 \in \Sigma^n$, let $\Delta(c_1, c_2)$ denote the Hamming distance between c_1 and c_2, that is, the number of positions in which c_1 and c_2 differ. Then the minimum distance d of the code C is defined as

$$d := \min_{\substack{m_1, m_2 \in \Sigma^k \\ m_1 \neq m_2}} \Delta[C(m_1), C(m_2)].$$

Since any code has to be injective, any two codewords will disagree in at least one position, so the distance of any code is at least 1. Besides, the **distance** equals the **minimum weight** for linear block codes because:

$$\min_{\substack{m_1, m_2 \in \Sigma^k \\ m_1 \neq m_2}} \Delta[C(m_1), C(m_2)] = \min_{\substack{m_1, m_2 \in \Sigma^k \\ m_1 \neq m_2}} \Delta[\mathbf{0}, C(m_1) + C(m_2)] = \min_{\substack{m \in \Sigma^k \\ m \neq \mathbf{0}}} w[C(m)] = w_{\min}.$$

A larger distance allows for more error correction and detection. For example, if we only consider errors that may change symbols of the sent codeword but never erase or add them, then the number of errors is the number of positions in which the sent codeword and the received word differ. A code with distance d allows the receiver to detect up to $d-1$ transmission errors since changing $d-1$ positions of a codeword can never accidentally yield another codeword. Furthermore, if no more than $(d-1)/2$ transmission errors occur, the receiver can uniquely decode the received word to a codeword. This is because every received word has at most one codeword at distance $(d-1)/2$. If more than $(d-1)/2$ transmission errors occur, the receiver cannot uniquely decode the received word in general as there might be several possible codewords. One way for the receiver to cope with this situation is to use list decoding, in which the decoder outputs a list of all codewords in a certain radius.

Popular notation

The notation $(n, k, d)_q$ describes a block code over an alphabet Σ of size q, with a block length n, message length k, and distance d. If the block code is a linear block code, then the square brackets in the notation $[n, k, d]_q$ are used to represent that fact. For binary codes with $q = 2$, the index is sometimes dropped. For maximum distance separable codes, the distance is always $d = n - k + 1$, but sometimes the precise distance is not known, non-trivial to prove or state, or not needed. In such cases, the d-component may be missing.

Sometimes, especially for non-block codes, the notation $(n, M, d)_q$ is used for codes that contain M codewords of length n. For block codes with messages of length k over an alphabet of size q, this number would be $M = q^k$.

Examples

As mentioned above, there are a vast number of error-correcting codes that are actually block codes. The first error-correcting code was the Hamming(7,4) code, developed by Richard W. Hamming in 1950. This code transforms a message consisting of 4 bits into a codeword of 7 bits by adding 3 parity bits. Hence this code is a block code. It turns out that it is also a linear code and that it has distance 3. In the shorthand notation above, this means that the Hamming(7,4) code is a $[7, 4, 3]_2$ code.

Reed–Solomon codes are a family of $[n,k,d]_q$ codes with $d=n-k+1$ and q being a prime power. Rank codes are family of $[n,k,d]_q$ codes with $d \leq n-k+1$. Hadamard codes are a family of $[n,k,d]_2$ codes with $n=2^{k-1}$ and $d=2^{k-2}$.

Error detection and correction properties

A codeword $c \in \Sigma^n$ could be considered as a point in the n-dimension space Σ^n and the code $\mathcal{C}$ is the subset of Σ^n. A code $\mathcal{C}$ has distance d means that $\forall c \in \mathcal{C}$, there is no other codeword in the *Hamming ball* centered at c with radius $d-1$, which is defined as the collection of n-dimension words whose *Hamming distance* to c is no more than $d-1$. Similarly, $\mathcal{C}$ with (minimum) distance d has the following properties:

- $\mathcal{C}$ can detect $d-1$ errors : Because a codeword c is the only codeword in the Hamming ball centered at itself with radius $d-1$, no error pattern of $d-1$ or fewer errors could change one codeword to another. When the receiver detects that the received vector is not a codeword of $\mathcal{C}$, the errors are detected (but no guarantee to correct).
- $\mathcal{C}$ can correct $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors. Because a codeword c is the only codeword in the Hamming ball centered at itself with radius $d-1$, the two Hamming balls centered at two different codewords respectively with both radius $\left\lfloor \frac{d-1}{2} \right\rfloor$ do not overlap with each other. Therefore, if we consider the error correction as finding the codeword closest to the received word y, as long as the number of errors is no more than $\left\lfloor \frac{d-1}{2} \right\rfloor$, there is only one codeword in the hamming ball centered at y with radius $\left\lfloor \frac{d-1}{2} \right\rfloor$, therefore all errors could be corrected.
- In order to decode in the presence of more than $(d-1)/2$ errors, list-decoding or maximum likelihood decoding can be used.
- $\mathcal{C}$ can correct $d-1$ erasures. By *erasure* it means that the position of the erased symbol is known. Correcting could be achieved by q-passing decoding : In i^{th} passing the erased position is filled with the i^{th} symbol and error correcting is carried out. There must be one passing that the number of errors is no more than $\left\lfloor \frac{d-1}{2} \right\rfloor$ and therefore the erasures could be corrected.

Lower and upper bounds of block codes

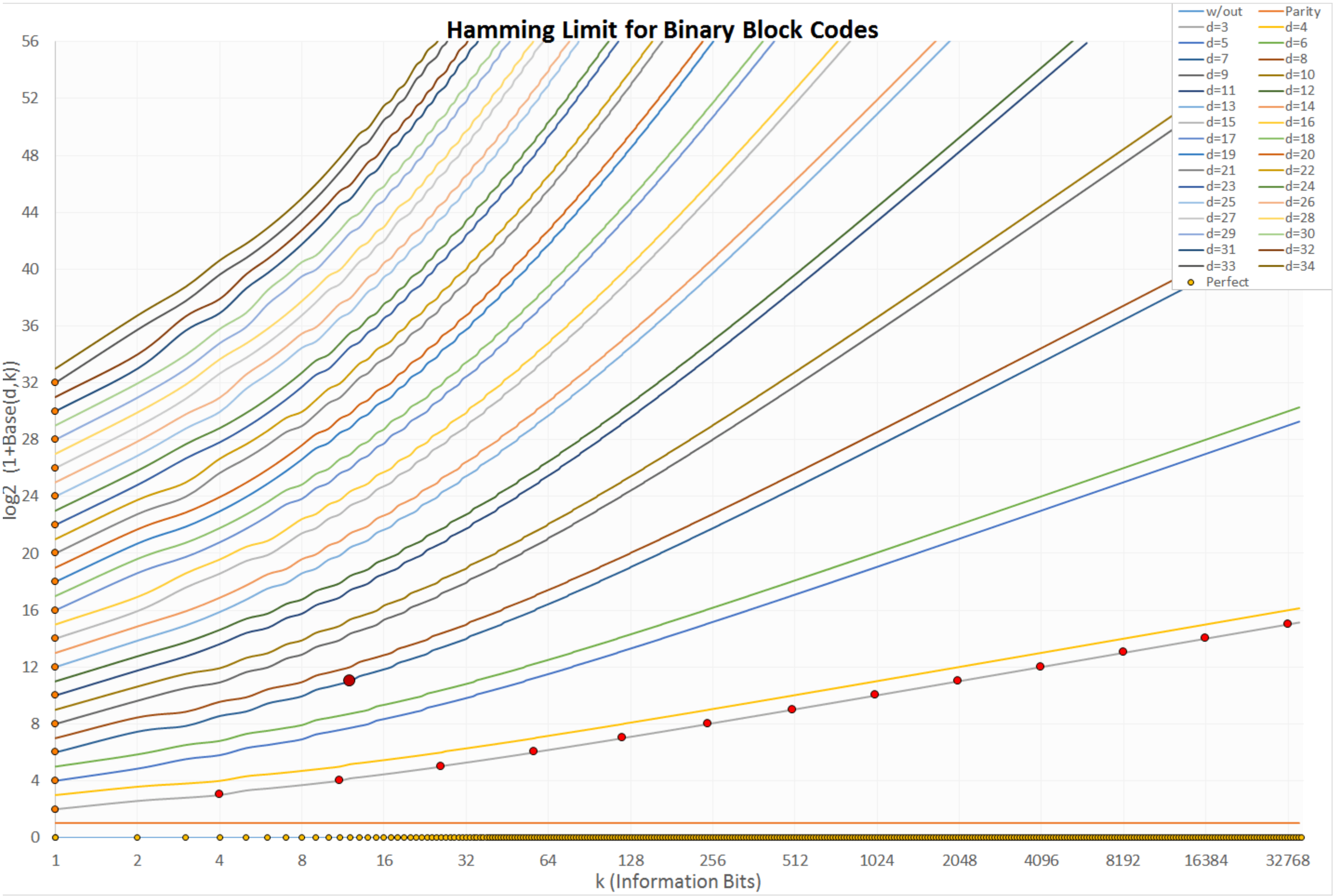
Family of codes

$C=\{C_i\}_{i\geq 1}$ is called *family of codes*, where C_i is an $(n_i,k_i,d_i)_q$ code with monotonic increasing n_i.

Rate of family of codes C is defined as $R(C)=\lim_{i\to\infty}\frac{k_i}{n_i}$

Relative distance of family of codes C is defined as $\delta(C)=\lim_{i\to\infty}\frac{d_i}{n_i}$

To explore the relationship between $R(C)$ and $\delta(C)$, a set of lower and upper bounds of block codes are known.



Hamming limit

Hamming bound

$$R \leq 1 - \frac{1}{n} \cdot \log_q \cdot \left[\sum_{i=0}^{\left\lfloor \frac{\delta \cdot n - 1}{2} \right\rfloor} \binom{n}{i} (q-1)^i \right]$$

Singleton bound

The Singleton bound is that the sum of the rate and the relative distance of a block code cannot be much larger than 1:

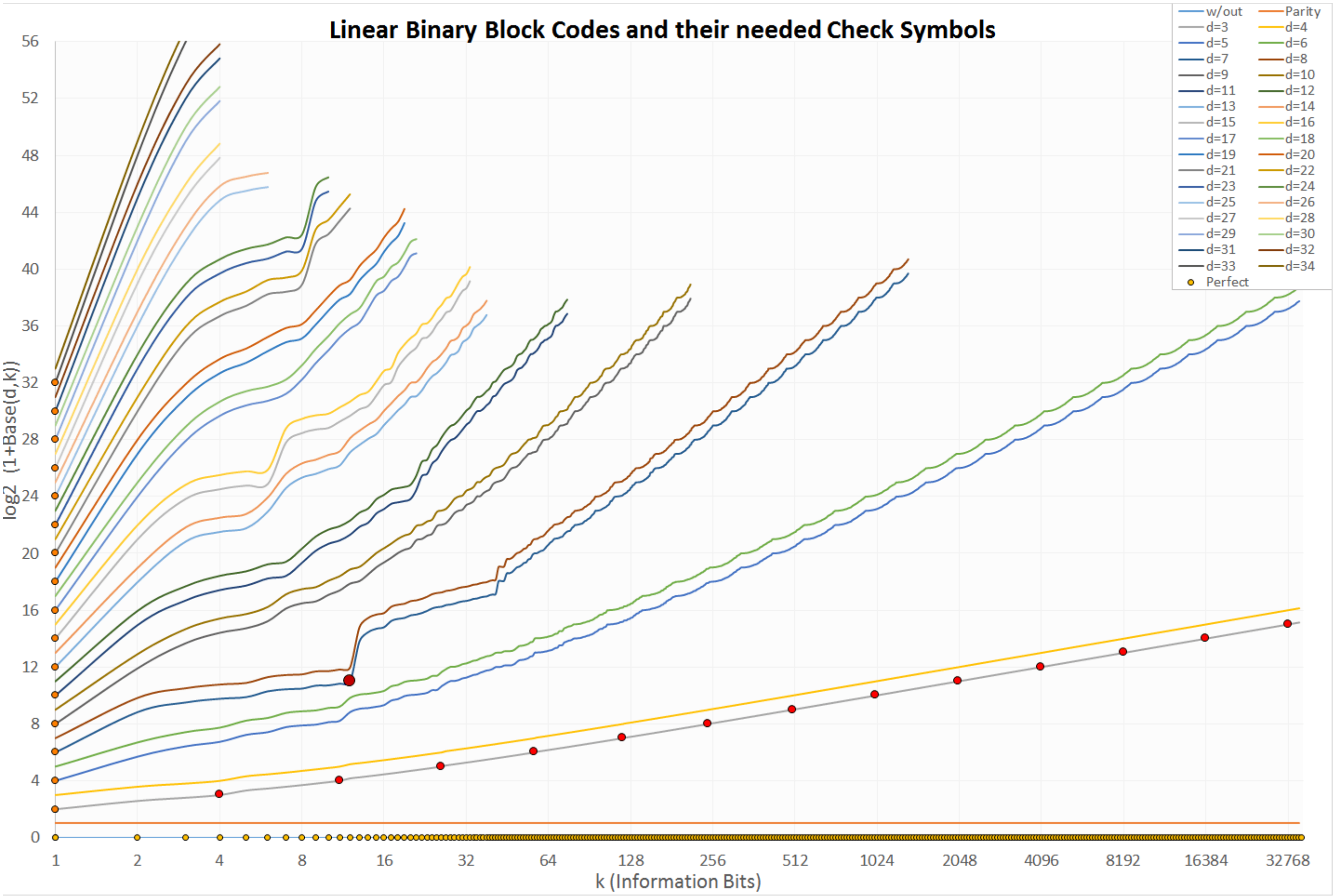
$$R + \delta \leq 1 + \frac{1}{n}.$$

In other words, every block code satisfies the inequality $k + d \leq n + 1$. Reed–Solomon codes are non-trivial examples of codes that satisfy the singleton bound with equality.

Plotkin bound

For $q = 2$, $R + 2\delta \leq 1$. In other words, $k + 2d \leq n$.

For the general case, the following Plotkin bounds holds for any $C \subseteq \mathbb{F}_q^n$ with distance d:



There are theoretical limits (such as the Hamming limit), but another question is which codes can actually constructed. It is like packing spheres in a box in many dimensions. This diagram shows the constructible codes, which are linear and binary. The *x* axis shows the number of protected symbols *k*, the *y* axis the number of needed check symbols *n*–*k*. Plotted are the limits for different Hamming distances from 1 (unprotected) to 34. Marked with dots are perfect codes:

- light orange on *x* axis: trivial unprotected codes
- orange on *y* axis: trivial repeat codes
- dark orange on data set *d*=3: classic perfect Hamming codes
- dark red and larger: the only perfect binary Golay code

1. If $d = \left(1 - \frac{1}{q}\right) n, |C| \le 2qn$
2. If $d > \left(1 - \frac{1}{q}\right) n, |C| \le \frac{qd}{qd - (q-1)\, n}$

For any q-ary code with distance δ, $R \le 1 - \left(\frac{q}{q-1}\right) \delta + o\,(1)$

Gilbert–Varshamov bound

$R \ge 1 - H_q\,(\delta) - \epsilon,$ where $0 \le \delta \le 1 - \frac{1}{q}, 0 \le \epsilon \le 1 - H_q\,(\delta),$

$H_q\,(x) \overset{\text{def}}{=} -x \cdot \log_q \frac{x}{q-1} - (1-x) \cdot \log_q\,(1-x)$ is the q-ary entropy function.

Johnson bound

Define $J_q\left(\delta\right) \overset{\text{def}}{=} \left(1-\frac{1}{q}\right)\left(1-\sqrt{1-\frac{q\delta}{q-1}}\right)$.

Let $J_q\left(n,d,e\right)$ be the maximum number of codewords in a Hamming ball of radius e for any code $C \subseteq \mathbb{F}_q^n$ of distance d.

Then we have the *Johnson Bound* : $J_q\left(n,d,e\right) \leq qnd$, if $\frac{e}{n} \leq \frac{q-1}{q}\left(1-\sqrt{1-\frac{q}{q-1}\cdot\frac{d}{n}}\right) = J_q\left(\frac{d}{n}\right)$

Elias–Bassalygo bound

$$R = \frac{\log_q\left|C\right|}{n} \leq 1 - H_q\left(J_q\left(\delta\right)\right) + o\left(1\right)$$

Sphere packings and lattices

Block codes are tied to the sphere packing problem which has received some attention over the years. In two dimensions, it is easy to visualize. Take a bunch of pennies flat on the table and push them together. The result is a hexagon pattern like a bee's nest. But block codes rely on more dimensions which cannot easily be visualized. The powerful Golay code used in deep space communications uses 24 dimensions. If used as a binary code (which it usually is), the dimensions refer to the length of the codeword as defined above.

The theory of coding uses the *N*-dimensional sphere model. For example, how many pennies can be packed into a circle on a tabletop or in 3 dimensions, how many marbles can be packed into a globe. Other considerations enter the choice of a code. For example, hexagon packing into the constraint of a rectangular box will leave empty space at the corners. As the dimensions get larger, the percentage of empty space grows smaller. But at certain dimensions, the packing uses all the space and these codes are the so-called perfect codes. There are very few of these codes.

Another property is the number of neighbors a single codeword may have.[1] Again, consider pennies as an example. First we pack the pennies in a rectangular grid. Each penny will have 4 near neighbors (and 4 at the corners which are farther away). In a hexagon, each penny will have 6 near neighbors. Respectively, in three and four dimensions, the maximum packing is given by the 12-face and 24-cell with 12 and 24 neighbors, respectively. When we increase the dimensions, the number of near neighbors increases very rapidly. In general, the value is given by the kissing numbers.

The result is that the number of ways for noise to make the receiver choose a neighbor (hence an error) grows as well. This is a fundamental limitation of block codes, and indeed all codes. It may be harder to cause an error to a single neighbor, but the number of neighbors can be large enough so the total error probability actually suffers.[1]

See also

- Channel capacity
- Shannon–Hartley theorem
- Noisy channel

- List decoding
- Sphere packing

References

1. Christian Schlegel and Lance Pérez (2004). *Trellis and turbo coding* (https://books.google.com/books?id=9wRCjfGAaEcC&pg=PA73). Wiley-IEEE. p. 73. ISBN 978-0-471-22755-7.

- J.H. van Lint (1992). *Introduction to Coding Theory* (https://archive.org/details/introductiontoco0000lint/page/31). GTM. **86** (2nd ed.). Springer-Verlag. p. 31 (https://archive.org/details/introductiontoco0000lint/page/31). ISBN 3-540-54894-7.
- F.J. MacWilliams; N.J.A. Sloane (1977). *The Theory of Error-Correcting Codes* (https://archive.org/details/theoryoferrorcor0000macw). North-Holland. p. 35 (https://archive.org/details/theoryoferrorcor0000macw/page/35). ISBN 0-444-85193-3.
- W. Huffman; V.Pless (2003). *Fundamentals of error-correcting codes* (https://archive.org/details/fundamentalsofer0000huff). Cambridge University Press. ISBN 978-0-521-78280-7.
- S. Lin; D. J. Jr. Costello (1983). *Error Control Coding: Fundamentals and Applications*. Prentice-Hall. ISBN 0-13-283796-X.

External links

- Charan Langton (2001) Coding Concepts and Block Coding (http://complextoreal.com/wp-content/uploads/2013/01/block.pdf)

Retrieved from "https://en.wikipedia.org/w/index.php?title=Block_code&oldid=931350894"

This page was last edited on 18 December 2019, at 11:36 (UTC).