Forward error correction

From Wikipedia, the free encyclopedia

In telecommunication, information theory, and coding theory, **forward error correction** (**FEC**) or **channel coding**[1] is a technique used for controlling errors in data transmission over unreliable or noisy communication channels. The central idea is the sender encodes the message in a redundant way by using an **error-correcting code** (ECC). The American mathematician Richard Hamming pioneered this field in the 1940s and invented the first error-correcting code in 1950: the Hamming (7,4) code.[2]

The redundancy allows the receiver to detect a limited number of errors that may occur anywhere in the message, and often to correct these errors without retransmission. FEC gives the receiver the ability to correct errors without needing a reverse channel to request retransmission of data, but at the cost of a fixed, higher forward channel bandwidth. FEC is therefore applied in situations where retransmissions are costly or impossible, such as one-way communication links and when transmitting to multiple receivers in multicast. FEC information is usually added to mass storage devices to enable recovery of corrupted data, and is widely used in modems.

FEC processing in a receiver may be applied to a digital bit stream or in the demodulation of a digitally modulated carrier. For the latter, FEC is an integral part of the initial analog-to-digital conversion in the receiver. The Viterbi decoder implements a soft-decision algorithm to demodulate digital data from an analog signal corrupted by noise. Many FEC coders can also generate a bit-error rate (BER) signal which can be used as feedback to fine-tune the analog receiving electronics.

The noisy-channel coding theorem establishes bounds on the theoretical maximum information transfer rate of a channel with some given noise level. Some advanced FEC systems come very close to the theoretical maximum.

The maximum fractions of errors or of missing bits that can be corrected is determined by the design of the FEC code, so different forward error correcting codes are suitable for different conditions.

Contents

How it works

FEC is accomplished by adding redundancy to the transmitted information using an algorithm. A redundant bit may be a complex function of many original information bits. The original information may or may not appear literally in the encoded output; codes that include the unmodified input in the output are **systematic**, while those that do not are **non-systematic**.

A simplistic example of FEC is to transmit each data bit 3 times, which is known as a (3,1) repetition code. Through a noisy channel, a receiver might see 8 versions of the output, see table below.

Triplet received	Interpreted as
000	0 (error free)
001	0
010	0
100	0
111	1 (error free)
110	1
101	1
011	1

This allows an error in any one of the three samples to be corrected by "majority vote" or "democratic voting". The correcting ability of this FEC is:

- Up to 1 bit of triplet in error, or
- up to 2 bits of triplet omitted (cases not shown in table).

Though simple to implement and widely used, this triple modular redundancy is a relatively inefficient FEC. Better FEC codes typically examine the last several dozen, or even the last several hundred, previously received bits to determine how to decode the current small handful of bits (typically in groups of 2 to 8 bits).

Averaging noise to reduce errors

FEC could be said to work by "averaging noise"; since each data bit affects many transmitted symbols, the corruption of some symbols by noise usually allows the original user data to be extracted from the other, uncorrupted received symbols that also depend on the same user data.

- Because of this "risk-pooling" effect, digital communication systems that use FEC tend to work well above a certain minimum signal-to-noise ratio and not at all below it.
- This *all-or-nothing tendency* — the cliff effect — becomes more pronounced as stronger codes are used that more closely approach the theoretical Shannon limit.
- Interleaving FEC coded data can reduce the all or nothing properties of transmitted FEC codes when the channel errors tend to occur in bursts. However, this method has limits; it is best used on narrowband data.

Most telecommunication systems use a fixed channel code designed to tolerate the expected worst-case bit error rate, and then fail to work at all if the bit error rate is ever worse. However, some systems adapt to the given channel error conditions: some instances of hybrid automatic repeat-request use a fixed FEC method as long as the FEC can handle the error rate, then switch to ARQ when the error rate gets too high; adaptive modulation and coding uses a variety of FEC rates, adding more error-correction bits per packet when there are higher error rates in the channel, or taking them out when they are not needed.

Types of FEC

The two main categories of FEC codes are block codes and convolutional codes.

- Block codes work on fixed-size blocks (packets) of bits or symbols of predetermined size. Practical block codes can generally be hard-decoded in polynomial time to their block length.
- Convolutional codes work on bit or symbol streams of arbitrary length. They are most often soft decoded with the Viterbi algorithm, though other algorithms are sometimes used. Viterbi decoding allows asymptotically optimal decoding efficiency with increasing constraint length of the convolutional code, but at the expense of exponentially increasing complexity. A convolutional code that is terminated is also a 'block code' in that it encodes a block of input data, but the block size of a convolutional code is generally arbitrary, while block codes have a fixed size dictated by their algebraic characteristics. Types of termination for convolutional codes include "tail-biting" and "bit-flushing".

There are many types of block codes, but among the classical ones the most notable is Reed-Solomon coding because of its widespread use on the Compact disc, the DVD, and in hard disk drives. Other examples of classical block codes include Golay, BCH, Multidimensional parity, and Hamming codes.

Hamming ECC is commonly used to correct NAND flash memory errors.[3] This provides single-bit error correction and 2-bit error detection. Hamming codes are only suitable for more reliable single level cell (SLC) NAND. Denser multi level cell (MLC) NAND requires stronger multi-bit correcting ECC such as

BCH or Reed–Solomon.[4][5] NOR Flash typically does not use any error correction.[4]

Classical block codes are usually decoded using **hard-decision** algorithms,[6] which means that for every input and output signal a hard decision is made whether it corresponds to a one or a zero bit. In contrast, convolutional codes are typically decoded using **soft-decision** algorithms like the Viterbi, MAP or BCJR algorithms, which process (discretized) analog signals, and which allow for much higher error-correction performance than hard-decision decoding.

Nearly all classical block codes apply the algebraic properties of finite fields. Hence classical block codes are often referred to as algebraic codes.

In contrast to classical block codes that often specify an error-detecting or error-correcting ability, many modern block codes such as LDPC codes lack such guarantees. Instead, modern codes are evaluated in terms of their bit error rates.

Most forward error correction correct only bit-flips, but not bit-insertions or bit-deletions. In this setting, the Hamming distance is the appropriate way to measure the bit error rate. A few forward error correction codes are designed to correct bit-insertions and bit-deletions, such as Marker Codes and Watermark Codes. The Levenshtein distance is a more appropriate way to measure the bit error rate when using such codes.[7]

Concatenated FEC codes for improved performance

Classical (algebraic) block codes and convolutional codes are frequently combined in **concatenated** coding schemes in which a short constraint-length Viterbi-decoded convolutional code does most of the work and a block code (usually Reed-Solomon) with larger symbol size and block length "mops up" any errors made by the convolutional decoder. Single pass decoding with this family of error correction codes can yield very low error rates, but for long range transmission conditions (like deep space) iterative decoding is recommended.

Concatenated codes have been standard practice in satellite and deep space communications since Voyager 2 first used the technique in its 1986 encounter with Uranus. The Galileo craft used iterative concatenated codes to compensate for the very high error rate conditions caused by having a failed antenna.

Low-density parity-check (LDPC)

Low-density parity-check (LDPC) codes are a class of recently re-discovered highly efficient linear block codes made from many single parity check (SPC) codes. They can provide performance very close to the channel capacity (the theoretical maximum) using an iterated soft-decision decoding approach, at linear time complexity in terms of their block length. Practical implementations rely heavily on decoding the constituent SPC codes in parallel.

LDPC codes were first introduced by Robert G. Gallager in his PhD thesis in 1960, but due to the computational effort in implementing encoder and decoder and the introduction of Reed–Solomon codes, they were mostly ignored until recently.

LDPC codes are now used in many recent high-speed communication standards, such as DVB-S2 (Digital video broadcasting), WiMAX (IEEE 802.16e standard for microwave communications), High-Speed Wireless LAN (IEEE 802.11n), 10GBase-T Ethernet (802.3an) and G.hn/G.9960 (ITU-T Standard for networking over power lines, phone lines and coaxial cable). Other LDPC codes are standardized for wireless communication standards within 3GPP MBMS (see fountain codes).

Turbo codes

Turbo coding is an iterated soft-decoding scheme that combines two or more relatively simple convolutional codes and an interleaver to produce a block code that can perform to within a fraction of a decibel of the Shannon limit. Predating LDPC codes in terms of practical application, they now provide similar performance.

One of the earliest commercial applications of turbo coding was the CDMA2000 1x (TIA IS-2000) digital cellular technology developed by Qualcomm and sold by Verizon Wireless, Sprint, and other carriers. It is also used for the evolution of CDMA2000 1x specifically for Internet access, 1xEV-DO (TIA IS-856). Like 1x, EV-DO was developed by Qualcomm, and is sold by Verizon Wireless, Sprint, and other carriers (Verizon's marketing name for 1xEV-DO is *Broadband Access*, Sprint's consumer and business marketing names for 1xEV-DO are *Power Vision* and *Mobile Broadband*, respectively).

Local decoding and testing of codes

Sometimes it is only necessary to decode single bits of the message, or to check whether a given signal is a codeword, and do so without looking at the entire signal. This can make sense in a streaming setting, where codewords are too large to be classically decoded fast enough and where only a few bits of the message are of interest for now. Also such codes have become an important tool in computational complexity theory, e.g., for the design of probabilistically checkable proofs.

Locally decodable codes are error-correcting codes for which single bits of the message can be probabilistically recovered by only looking at a small (say constant) number of positions of a codeword, even after the codeword has been corrupted at some constant fraction of positions. Locally testable codes are error-correcting codes for which it can be checked probabilistically whether a signal is close to a codeword by only looking at a small number of positions of the signal.

Interleaving

Interleaving is frequently used in digital communication and storage systems to improve the performance of forward error correcting codes. Many communication channels are not memoryless: errors typically occur in bursts rather than independently. If the number of errors within a code word exceeds the error-correcting code's capability, it fails to recover the original code word. Interleaving ameliorates this problem by shuffling source symbols across several code words, thereby creating a more uniform distribution of errors.[8] Therefore, interleaving is widely used for burst error-correction.

The analysis of modern iterated codes, like turbo codes and LDPC codes, typically assumes an independent distribution of errors.[9] Systems using LDPC codes therefore typically employ additional interleaving across the symbols within a code word.[10]

For turbo codes, an interleaver is an integral component and its proper design is crucial for good performance.[8][11] The iterative decoding algorithm works best when there are not short cycles in the factor graph that represents the decoder; the interleaver is chosen to avoid short cycles.

Interleaver designs include:

- rectangular (or uniform) interleavers (similar to the method using skip factors described above)
- convolutional interleavers
- random interleavers (where the interleaver is a known random permutation)
- S-random interleaver (where the interleaver is a known random permutation with the constraint that no input symbols within distance S appear within a distance of S in the output).[12]
- Another possible construction is a contention-free quadratic permutation polynomial (QPP).[13] It is used for example in the 3GPP Long Term Evolution mobile telecommunication standard.[14]

In multi-carrier communication systems, interleaving across carriers may be employed to provide frequency diversity, e.g., to mitigate frequency-selective fading or narrowband interference.[15]

Example

Transmission without interleaving:

```
Error-free message:                                 aaaabbbbccccddddeeeeffffgggg
Transmission with a burst error:                    aaaabbbbccc____deeeeffffgggg
```

Here, each group of the same letter represents a 4-bit one-bit error-correcting codeword. The codeword cccc is altered in one bit and can be corrected, but the codeword dddd is altered in three bits, so either it cannot be decoded at all or it might be decoded incorrectly.

With interleaving:

```
Error-free code words:                              aaaabbbbccccddddeeeeffffgggg
Interleaved:                                        abcdefgabcdefgabcdefgabcdefg
Transmission with a burst error:                    abcdefgabcd____bcdefgabcdefg
Received code words after deinterleaving:           aa_abbbbccccdddde_eef_ffg_gg
```

In each of the codewords aaaa, eeee, ffff, gggg, only one bit is altered, so one-bit error-correcting code will decode everything correctly.

Transmission without interleaving:

```
Original transmitted sentence:                      ThisIsAnExampleOfInterleaving
Received sentence with a burst error:               ThisIs______pleOfInterleaving
```

The term "AnExample" ends up mostly unintelligible and difficult to correct.

With interleaving:

```
Transmitted sentence:                               ThisIsAnExampleOfInterleaving...
```

```
Error-free transmission:                         TIEpfeaghsxlIrv.iAaenli.snmOten.
Received sentence with a burst error:            TIEpfe______Irv.iAaenli.snmOten.
Received sentence after deinterleaving:          T_isI_AnE_amp_eOfInterle_vin_...
```

No word is completely lost and the missing letters can be recovered with minimal guesswork.

Disadvantages of interleaving

Use of interleaving techniques increases total delay. This is because the entire interleaved block must be received before the packets can be decoded.[16] Also interleavers hide the structure of errors; without an interleaver, more advanced decoding algorithms can take advantage of the error structure and achieve more reliable communication than a simpler decoder combined with an interleaver.

List of error-correcting codes

Distance	Code
2 (single-error detecting)	Parity
3 (single-error correcting)	Triple modular redundancy
3 (single-error correcting)	perfect Hamming such as Hamming(7,4)
4 (SECDED)	Extended Hamming
5 (double-error correcting)	
6 (double-error correct-/triple error detect)	
7 (three-error correcting)	perfect binary Golay code
8 (TECFED)	extended binary Golay code

- AN codes
- BCH code, which can be designed to correct any arbitrary number of errors per code block.
- Berger code
- Constant-weight code
- Convolutional code
- Expander codes
- Group codes
- Golay codes, of which the Binary Golay code is of practical interest
- Goppa code, used in the McEliece cryptosystem
- Hadamard code
- Hagelbarger code
- Hamming code
- Latin square based code for non-white noise (prevalent for example in broadband over powerlines)
- Lexicographic code
- Long code
- Low-density parity-check code, also known as Gallager code, as the archetype for sparse graph codes
- LT code, which is a near-optimal rateless erasure correcting code (Fountain code)
- m of n codes
- Online code, a near-optimal rateless erasure correcting code
- Polar code (coding theory)
- Raptor code, a near-optimal rateless erasure correcting code

- Reed–Solomon error correction
- Reed–Muller code
- Repeat-accumulate code
- Repetition codes, such as Triple modular redundancy
- Spinal code, a rateless, nonlinear code based on pseudo-random hash functions [17]
- Tornado code, a near-optimal erasure correcting code, and the precursor to Fountain codes
- Turbo code
- Walsh–Hadamard code

See also

- Code rate
- Erasure codes
- Soft-decision decoder
- Error detection and correction
- Error-correcting codes with feedback
- Burst error-correcting code

References

1. Charles Wang, Dean Sklar, Diana Johnson (Winter 2001–2002). "Forward Error-Correction Coding". *Crosslink — The Aerospace Corporation magazine of advances in aerospace technology* (The Aerospace Corporation) **3** (1). "How Forward Error-Correcting Codes Work"
2. Hamming, R. W. (April 1950). "Error Detecting and Error Correcting Codes" (PDF). *Bell System Tech. J.* (USA: AT&T) **29** (2): 147–160. doi:10.1002/j.1538-7305.1950.tb00463.x. Retrieved 4 December 2012.
3. "Hamming codes for NAND flash memory devices" (http://www.eetasia.com/ART_8800575062_499486_AN_7549c493.HTM). EE Times-Asia. Apparently based on "Micron Technical Note TN-29-08: Hamming Codes for NAND Flash Memory Devices" (http://www.micron.com/~/media/Documents/Products/Technical%20Note/NAND%20Flash/tn2908_NAND_hamming_ECC_code.pdf). 2005. Both say: "The Hamming algorithm is an industry-accepted method for error detection and correction in many SLC NAND flash-based applications."
4. "What Types of ECC Should Be Used on Flash Memory?" (http://www.spansion.com/Support/Application%20Notes/Types_of_ECC_Used_on_Flash_AN.pdf). (Spansion application note). 2011. says: "Both Reed-Solomon algorithm and BCH algorithm are common ECC choices for MLC NAND flash. ... Hamming based block codes are the most commonly used ECC for SLC.... both Reed-Solomon and BCH are able to handle multiple errors and are widely used on MLC flash."
5. Jim Cooke. "The Inconvenient Truths of NAND Flash Memory" (http://cushychicken.github.io/assets/cooke_inconvenient_truths.pdf). 2007. p. 28. says "For SLC, a code with a correction threshold of 1 is sufficient. t=4 required ... for MLC."
6. Baldi M., Chiaraluce F. (2008). "A Simple Scheme for Belief Propagation Decoding of BCH and RS Codes in Multimedia Transmissions". *International Journal of Digital Multimedia Broadcasting* **2008**: 957846. doi:10.1155/2008/957846.
7. Shah, Gaurav; Molina, Andres; Blaze, Matt (2006). "Keyboards and covert channels" (PDF). *Proceedings of the 15th conference on USENIX Security Symposium*.
8. B. Vucetic, J. Yuan (2000). *Turbo codes: principles and applications*. Springer Verlag. ISBN 978-0-7923-7868-6.
9. M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, V. Stemann (1997). "Practical Loss-Resilient Codes". *Proc. 29th annual Association for Computing Machinery (ACM) symposium on Theory of computation*.
10. "Digital Video Broadcast (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other satellite broadband applications (DVB-S2)". *En 302 307* (ETSI) (V1.2.1). April 2009.

11. K. Andrews; et al. (November 2007). "The Development of Turbo and LDPC Codes for Deep-Space Applications". *Proc. of the IEEE* **95** (11).
12. S. Dolinar and D. Divsalar. Weight Distributions for Turbo Codes Using Random and Nonrandom Permutations. 1995. [1] (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.105.6640&rep=rep1&type=pdf)
13. Takeshita, Oscar (2006). "Permutation Polynomial Interleavers: An Algebraic-Geometric Perspective". arXiv:cs/0601048.
14. 3GPP TS 36.212 (http://www.3gpp.org/ftp/Specs/html-info/36212.htm), version 8.8.0, page 14
15. "Digital Video Broadcast (DVB); Frame structure, channel coding and modulation for a second generation digital terrestrial television broadcasting system (DVB-T2)". *En 302 755* (ETSI) (V1.1.1). September 2009.
16. "Explaining Interleaving - W3techie". w3techie.com. Retrieved 2010-06-03.
17. Perry, Jonathan; Balakrishnan, Hari; Shah, Devavrat (2011). "Rateless Spinal Codes". *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*. doi:10.1145/2070562.2070568.

Further reading

- Clark, George C., Jr.; Cain, J. Bibb (1981). *Error-Correction Coding for Digital Communications*. New York: Plenum Press. ISBN 0-306-40615-2.
- Wicker, Stephen B. (1995). *Error Control Systems for Digital Communication and Storage*. Englewood Cliffs NJ: Prentice-Hall. ISBN 0-13-200809-2.
- Wilson, Stephen G. (1996). *Digital Modulation and Coding*. Englewood Cliffs NJ: Prentice-Hall. ISBN 0-13-210071-1.
- "Error Correction Code in Single Level Cell NAND Flash memories" (http://www.st.com/stonline/products/literature/an/10123.htm) 16 February 2007
- "Error Correction Code in NAND Flash memories" (http://www.eetasia.com/ARTICLES/2004NOV/A/2004NOV29_MEM_AN10.PDF?SOURCES=DOWNLOAD) 29 November 2004
- Observations on Errors, Corrections, & Trust of Dependent Systems (http://perspectives.mvdirona.com/2012/02/observations-on-errors-corrections-trust-of-dependent-systems/), by James Hamilton, February 26, 2012

External links

- Morelos-Zaragoza, Robert (2004). "The Error Correcting Codes (ECC) Page". Retrieved 2006-03-05.

Retrieved from "https://en.wikipedia.org/w/index.php?title=Forward_error_correction&oldid=706342964"

Categories: Error detection and correction