Good Error–Correcting Codes based on Very Sparse Matrices

David J.C. MacKay
Cavendish Laboratory, Cambridge, CB3 0HE.
United Kindom. mackay@mrao.cam.ac.uk

June 9, 1997— submitted to IEEE–IT

Abstract

We present two families of error-correcting codes defined in terms of very sparse matrices. 'MN codes' are new, and 'GL codes' were first investigated by Gallager in 1962, but appear to have been largely forgotten, in spite of their excellent properties. The decoding of both codes can be tackled with a practical belief propagation algorithm.

We prove that these codes are 'very good', in that sequences of codes exist which, when optimally decoded, achieve information rates up to the Shannon limit. This result holds not only for the binary symmetric channel but also for any channel with symmetric stationary ergodic noise.

We give experimental results for binary symmetric channels and Gaussian channels demonstrating that practical performance substantially better than that of standard convolutional and concatenated codes can be achieved; indeed the performance of GL codes is almost as close to the Shannon limit as that of Turbo codes.

Keywords

Error-correction codes, iterative probabilistic decoding, Shannon Limit.

1 Introduction

For a glossary of symbols used in this paper, please see appendix A.

1.1 Background

In 1948, Shannon [53] proved that for any channel there exist block codes that achieve arbitrarily small probability of error at any communication rate up to the capacity of the channel. We will refer to such code families as 'very good' codes. By 'good' codes we mean code families that achieve arbitrarily small probability of error at non-zero communication rates up to some maximum rate that may be *less than* the capacity of the given channel. By 'bad' codes we mean code families that can only achieve arbitrarily small probability of error by decreasing the information rate to zero. (Bad codes are not necessarily useless for practical purposes.) By 'practical' codes we mean code families which can be encoded and decoded in time and space polynomial in the block length.

Shannon's proof was non-constructive and employed random codes for which there is no practical encoding or decoding algorithm. Since 1948, it has been proved that there exist very good linear codes (non-constructively); that there exist very good cyclic codes (non-constructively) [40]; that there exist very good codes that have structure (non-constructively) [17]; and that very good codes can be produced with a short description in terms of permutations [1]. But no practical decoding algorithm is known for any of these codes, and it is known that the general linear decoding problem

(find the maximum likelihood $\mathbf{s}$ in the equation $\mathbf{G}^{\mathrm{T}}\mathbf{s} + \mathbf{n} = \mathbf{r}$) is NP–complete [10]. Convolutional codes (which can be viewed as block codes with memory) can approach the Shannon limit as their constraint length increases but the complexity of their decoding grows exponentially with the constraint length. For a long time a generally held view was that for practical purposes a channel's effective capacity was a rate 'R_0' smaller than the Shannon capacity, if convolutional codes were used; and many believed this conjecture applied to all codes, speculating that practical communication beyond R_0 was impossible. Forney proved that there do exist very good 'concatenated' codes that are practical [21]; but the proof was also non-constructive [40].

When it comes to practical, *constructive* codes, constructions have been demonstrated of codes based on concatenation that are good, though not very good, but most known practical codes are asymptotically bad [40]. Goppa's recent algebraic geometry codes (reviewed in [60]) appear to be both practical and good (with practical decoding proven possible up to the Gilbert bound), but we believe that the literature has not established whether they are very good. The most practical decoding algorithm [20] appears to be prohibitively costly (N^3) to implement, and algebraic geometry codes do not appear to be destined for practical use.

Thus the conventional view is that there are few known constructive codes that are good, fewer still that are practical, and none at all that are both practical and very good. It seems to be widely believed that whereas almost any random linear code is good, codes with structure that allows practical coding are likely to be bad [40], [14]. Battail expresses an alternative view, however, that 'we can think of good codes, and we can decode them' [6]. This statement is supported by the results of the present paper.

In this paper we present two code families. Gallager's low-density parity-check codes ('GL codes') are defined in terms of a very sparse random parity check matrix [23, 24, 37]. 'MN codes' are also defined in terms of very sparse random matrices, and were first presented in [36]. (We generalized MN codes to GL codes, then realised that we had rediscovered Gallager's work.) MN codes are unconventional in that redundancy can be incorporated in the transmitted codewords not only by using a $K \times N$ generator matrix with transmitted block length N greater than the source block length K, but also by using a source that is itself redundant.

These code families both have two important properties. First, because of the construction in terms of sparse matrices, practical decoding seems to be possible at good communication rates. Second, we prove in section 2 that in spite of their simple construction these codes are *very good* — that is, sequences of codes exist which when optimally decoded achieve information rates up to the Shannon limit of the binary symmetric channel. We further prove that the same codes are in fact good for any ergodic symmetric channel model. Our proof may be viewed as a semi-constructive proof of Shannon's noisy channel coding theorem. It is indeed easy to think of good codes.

In section 3 we present a 'belief propagation' algorithm for solving the decoding problem, first presented by Gallager [23]. We give an analysis of the decoding algorithm in section 3.3. These results lead us to conjecture that there exist GL and MN codes which are not only good but which also achieve error rates approaching zero at a non-zero information rate when decoded using a practical algorithm. In sections 4 and 6 we describe empirical results of computer experiments using the belief propagation algorithm to decode GL and MN codes. Our experiments show that practical performance significantly superior to that of textbook codes can be achieved by these codes. Section 6 contains discussion specific to MN codes.

1.2 Definitions

The input and output alphabets of the binary symmetric channel (BSC) will be denoted $\{0, 1\}$. We will denote the error probability of the BSC by f_{n}, where $f_{\mathrm{n}} < 0.5$.

Definition 1 *The binary entropy functions $H_2(f)$ and $H_2^e(f)$ are*

$$H_2(f) \quad = \quad f \log_2(1/f) + (1-f) \log_2(1/(1-f)) \tag{1}$$

$$H_2^e(f) \quad = \quad f \log_e (1/f) + (1-f) \log_e (1/(1-f)). \tag{2}$$

Definition 2 *The* weight *of a binary vector or matrix is the number of 1s in it. We denote the weight of a vector* $\mathbf{x}$ *by* $w(\mathbf{x})$. *The* density *of a source of random bits is the expected fraction of 1 bits. A source is* sparse *if its density is less than 0.5. A vector* $\mathbf{v}$ *is* very sparse *if its density vanishes as its length increases, for example, if a constant number* t *of its bits are 1s. The* overlap *between two vectors is the number of 1s in common between them.*

Definition 3 *The* capacity $C(f_{\rm n})$ *of a binary symmetric channel with noise density* $f_{\rm n}$ *is, in bits per cycle,*

$$C(f_{\rm n}) = 1 - H_2(f_{\rm n}). \tag{3}$$

The rate $R_0(f_{\rm n})$ *is*

$$R_0(f_{\rm n}) \equiv 1 - \log_2 \left[1 + 2\sqrt{f_{\rm n}(1-f_{\rm n})} \right]. \tag{4}$$

This is the computational cutoff of sequential decoding for convolutional codes—the rate beyond which the expected cost of achieving vanishing error probability using sequential decoding becomes infinite.

The Gilbert bound $GV(f_{\rm n})$ *is*

$$GV(f_{\rm n}) = \begin{cases} 1 - H_2(2f_{\rm n}) & f_{\rm n} < 1/4 \\ 0 & f_{\rm n} \geq 1/4 \end{cases}. \tag{5}$$

This is the maximum rate at which one can communicate with a code whose codewords satisfy the Gilbert–Varshamov minimum distance bound, assuming bounded distance decoding [38].

Definition 4 *A model that defines a probability distribution over strings* $\mathbf{x}$ *of any length* N, $P(\mathbf{x}|N)$, *has* mean entropy H_x *if for any* $\epsilon > 0$ *and* $\eta > 0$ *there exists an* N^* *such that for all* $N > N^*$,

$$P\left(\left| \frac{1}{N} \log_2 \frac{1}{P(\mathbf{x}|N)} - H_x \right| > \eta \right) < \epsilon. \tag{6}$$

For example, a memoryless binary symmetric channel's noise has mean entropy $H_{\rm n} = H_2(f_{\rm n})$, where $f_{\rm n}$ is the density of the noise; the proof of this statement, by the law of large numbers, is well known [15]. We will prove that the codes presented in this paper are good codes not only for the binary symmetric channel but also a wide class of channels with memory.

Definition 5 *A* binary channel with symmetric stationary ergodic noise *is a channel whose output in response to a transmitted binary vector* $\mathbf{t}$ *is given by* $\mathbf{r} = \mathbf{t} + \mathbf{n}$ *mod* 2, *where* $\mathbf{n}$, *the noise vector, has a probability distribution that is (a) independent of* $\mathbf{t}$ *and (b) stationary and ergodic.*

For example, burst noise might be modelled by a stationary and ergodic Markov process. Such a process has a mean entropy, though the evaluation of this quantity may be challenging. The standard Gaussian channel with binary inputs is also equivalent to a binary channel with stationary stationary ergodic noise.

We will concentrate on the case of a binary channel with symmetric noise (see definition 5) in the body of this paper. Channels with memory whose inputs are binary and whose outputs are in some more general alphabet are addressed in appendix H.

1.2.1 Linear codes

A linear error correcting code can be represented by a N by K binary matrix $\mathbf{G}^{\mathrm{T}}$ (the **generator matrix**), such that a K-bit binary message $\mathbf{s}$ is encoded as the N-bit vector $\mathbf{t} = \mathbf{G}^{\mathrm{T}}\mathbf{s} \bmod 2$. (Note that we have chosen to use column vectors so the generator matrices act to the right rather than the left.) The generator matrix is in 'systematic form' if it can be written as

$$\mathbf{G}^{\mathrm{T}} = \begin{bmatrix} \mathbf{I}_K \\ \mathbf{P} \end{bmatrix}, \tag{7}$$

where $\mathbf{I}_K$ is the $K \times K$ identity matrix, and $\mathbf{P}$ is a binary matrix. The channel adds noise $\mathbf{n}$ to the vector $\mathbf{t}$ with the resulting received signal $\mathbf{r}$ being given by:

$$\mathbf{r} = (\mathbf{G}^{\mathrm{T}}\mathbf{s} + \mathbf{n}) \bmod 2. \tag{8}$$

The decoder's task is to infer $\mathbf{s}$ given the received message $\mathbf{r}$, and the assumed noise properties of the channel. The *optimal decoder* returns the message $\mathbf{s}$ that maximizes the posterior probability

$$P(\mathbf{s}|\mathbf{r}, \mathbf{G}) = \frac{P(\mathbf{r}|\mathbf{s}, \mathbf{G})P(\mathbf{s})}{P(\mathbf{r}|\mathbf{G})}. \tag{9}$$

It is often not practical to implement the optimal decoder.

If the prior probability of $\mathbf{s}$ is assumed uniform, and the probability of $\mathbf{n}$ is assumed to be independent of $\mathbf{s}$ (c.f. definition 5), then it is convenient to introduce the $(N - K) \times N$ **parity check matrix**, $\mathbf{H}$, which in systematic form is $[\mathbf{P}\ \mathbf{I}_{N-K}]$. The parity check matrix has the property $\mathbf{H}\mathbf{G}^{\mathrm{T}} = 0 \bmod 2$, so that, applying $\mathbf{H}$ to equation (8),

$$\mathbf{H}\mathbf{n} = \mathbf{H}\mathbf{r} \bmod 2. \tag{10}$$

Any other $(N - K) \times N$ matrix $\mathbf{A}$ whose rows span the same space as $\mathbf{H}$ is a valid parity check matrix.

The decoding problem thus reduces, given the above assumptions, to the task of finding the most probable noise vector $\mathbf{n}$ such that

$$\mathbf{H}\mathbf{n} \bmod 2 = \mathbf{z}, \tag{11}$$

where the syndrome vector $\mathbf{z} = \mathbf{H}\mathbf{r} \bmod 2$.

1.3 Description of the two code families

We define two code families. We explain the more conventional GL codes first.

1.3.1 The idea behind GL codes

We construct a linear code by first making a very sparse random parity check matrix $\mathbf{A}$. (Very sparse, but *not* systematic.) We then use linear algebra to obtain a corresponding generator matrix. This can be done by first putting $\mathbf{A}$ into systematic form $\mathbf{H}$, then deducing a systematic $\mathbf{G}$. This simple construction is complemented by a straightforward decoding algorithm which implements an approximation to the ideal Bayesian inference of equation (9).

1.3.2 Construction of GL codes

The parity check matrix $\mathbf{A}$ can be constructed as follows. We will describe variations on this construction later.

A transmitted block length N and a source block length K are selected. We define $M = N - K$ to be the number of parity checks. We select a *column weight* t, which is an integer greater than or equal to 3. We create a rectangular $M \times N$ matrix [M rows and N columns] $\mathbf{A}$ at random with exactly weight t per column and a weight per row as uniform as possible. If N/M is chosen to be an appropriate ratio of integers then the number per row can be constrained to be exactly tN/M. We then use Gaussian elimination and reordering of columns to derive an equivalent parity check matrix in systematic form $[\mathbf{P}\,\mathbf{I}_M]$. There is a possibility that the rows of $\mathbf{A}$ are not independent (though for odd t, this has small probability); in this case, $\mathbf{A}$ is a parity check matrix for a code with the same N and with smaller M, that is, a code with *greater* rate than assumed in the following sections. Redefining $\mathbf{A}$ to be the original matrix with its columns reordered as in the Gaussian elimination, we have the following situation.

The matrix $\mathbf{A} = [\mathbf{C}_1\mathbf{C}_2]$ is composed of two very sparse matrices $\mathbf{C}_1$ and $\mathbf{C}_2$ as follows.

The matrix $\mathbf{C}_2$ is a square $M \times M$ matrix that is very sparse and invertible. The inverse $\mathbf{C}_2^{-1}$ of this matrix has been computed during the Gaussian elimination which produced the matrix $\mathbf{P} = \mathbf{C}_2^{-1}\mathbf{C}_1$. The inversion takes order M^2N time and is performed once only.

The matrix $\mathbf{C}_1$ is a rectangular $M \times K$ matrix that is very sparse.

Encoding. We define the generator matrix of the GL code to be

$$\mathbf{G}^{\mathrm{T}} = \begin{bmatrix} \mathbf{I}_K \\ \mathbf{P} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_K \\ \mathbf{C}_2^{-1}\mathbf{C}_1 \end{bmatrix} \bmod 2, \tag{12}$$

where $\mathbf{I}_K$ is the $K \times K$ identity matrix.

1.3.3 Variations

1. When generating the matrix $\mathbf{A}$, one can constrain all pairs of columns in the matrix to have an overlap ≤ 1. This is expected to improve the properties of the ensemble of codes, for reasons that will become apparent in section 2.2.

2. One can further constrain the matrix $\mathbf{A}$ so that the topology of the corresponding bipartite graph does not contain short cycles. This is discussed further in section 4.3.

1.3.4 The decoding problem for GL codes

A source vector $\mathbf{s}$ of length K is encoded into a transmitted vector $\mathbf{t}$ defined by:

$$\mathbf{t} = \mathbf{G}^{\mathrm{T}}\mathbf{s} \bmod 2. \tag{13}$$

If the generator matrix has been computed explicitly (which takes M^2N time) then the transmitted vector can be computed by explicit multiplication in NK time. However, encoding might be possible in less time using sparse matrix methods.

The received vector is

$$\mathbf{r} = \mathbf{t} + \mathbf{n} \bmod 2, \tag{14}$$

where the noise is $\mathbf{n}$. In the case of a binary symmetric channel, $\mathbf{n}$ is assumed to be a sparse random vector with independent, identically-distributed bits of density f_{n}. We will discuss more general channels below.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.