Near Shannon Limit Performance of Low Density Parity Check Codes

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

Radford M. Neal
Depts. of Statistics and Computer Science, Univ. of Toronto,
M5S 1A4, Canada. radford@stat.toronto.edu

July 12, 1996—To be published in Electronics Letters

Indexing terms: Error-correction codes, probabilistic decoding.

Abstract

We report the empirical performance of Gallager's low density parity check codes on Gaussian channels. We show that performance substantially better than that of standard convolutional and concatenated codes can be achieved; indeed the performance is almost as close to the Shannon limit as that of Turbo codes.

A linear code may be described in terms of a generator matrix $\mathbf{G}$ or in terms of a parity check matrix $\mathbf{H}$, which satisfies $\mathbf{H}\mathbf{x} = 0$ for all codewords $\mathbf{x}$. In 1962, Gallager reported work on binary codes defined in terms of low density parity check matrices (abbreviated 'GL codes') [5, 6]. The matrix $\mathbf{H}$ was defined in a non-systematic form; each column of $\mathbf{H}$ had a small weight (*e.g.*, 3) and the weight per row was also uniform; the matrix $\mathbf{H}$ was constructed at random subject to these constraints. Gallager proved distance properties of these codes and described a probability-based decoding algorithm with promising empirical performance. However it appears that GL codes have been generally forgotten, the assumption perhaps being that concatenated codes [4] were superior for practical purposes (R.G. Gallager, personal communication).

During our work on MN codes [8] we realised that it is possible to create 'good' codes from very sparse random matrices, and to decode them (even beyond their minimum distance) using approximate probabilistic algorithms. We eventually reinvented Gallager's decoding algorithm and GL codes. In this paper we report the empirical performance of these codes on Gaussian channels. We have proved theoretical properties of GL codes (essentially, that the channel coding theorem holds for them) elsewhere [9]. GL codes can also be defined over $GF(q)$. We are currently implementing this generalization.

We created sparse random parity check matrices in the following ways.

Construction 1A. An M by N matrix (M rows, N columns) is created at random with weight per column t (*e.g.*, $t = 3$), and weight per row as uniform as possible, and overlap between any two columns no greater than 1. (The weight of a column is the number of non-zero elements; the overlap between two columns is their inner product.)

Construction 2A. Up to $M/2$ of the columns are designated weight 2 columns, and these are constructed such that there is zero overlap between any pair of columns. The remaining columns are made at random with weight 3, with the weight per row as uniform as possible, and overlap between any two columns of the entire matrix no greater than 1.

Constructions 1B and 2B. A small number of columns are deleted from a matrix produced by constructions 1A and 2A, respectively, so that the bipartite graph corresponding to the matrix has no short cycles of length less than some length l.

The above constructions do not ensure that all the rows of the matrix are linearly independent, so the $M \times N$ matrix created is the parity matrix of a linear code with rate *at least* $R \equiv K/N$, where $K = N - M$. We report results on the assumption that the rate is R. The generator matrix of the code can be created by Gaussian elimination.

We simulated a Gaussian channel with binary input $\pm a$ and additive noise of variance $\sigma^2 = 1$. If one communicates using a code of rate R then it is conventional to describe the signal to noise ratio by $\frac{E_b}{N_0} = \frac{a^2}{2R\sigma^2}$ and to report this number in decibels as $10 \log_{10} E_b/N_0$.

Decoding. The decoding problem is to find the most probable vector $\mathbf{x}$ such that $\mathbf{Hx} \bmod 2 = 0$, with the likelihood of $\mathbf{x}$ given by $\prod_n f_n^{x_n}$ where $f_n^1 = 1/(1+\exp(-2ay_n/\sigma^2))$ and $f_n^0 = 1 - f_n^1$, and y_n is the channel's output at time n.

Gallager's algorithm (reviewed in detail in [9]) may be viewed as an approximate belief propagation algorithm [10]. (The Turbo decoding algorithm may also be viewed as a belief propagation algorithm (R.J.McEliece and D.J.C.MacKay, unpublished).)

We refer to the elements of $\mathbf{x}$ as bits and to the rows of $\mathbf{H}$ as checks. We denote the set of bits n that participate in check m by $\mathcal{N}(m) \equiv \{n : H_{mn} = 1\}$. Similarly we define the set of checks in which bit n participates, $\mathcal{M}(n) \equiv \{m : H_{mn} = 1\}$. We denote a set $\mathcal{N}(m)$ with bit n excluded by $\mathcal{N}(m)\backslash n$. The algorithm has two alternating parts, in which quantities q_{mn} and r_{mn} associated with each non-zero element in the $\mathbf{H}$ matrix are iteratively updated. The quantity q_{mn}^x is meant to be the probability that bit n of $\mathbf{x}$ is x, given the information obtained via checks other than check m. The quantity r_{mn}^x is meant to be the probability of check m being satisfied if bit n of $\mathbf{x}$ is considered fixed at x and the other bits have a separable distribution given by the probabilities $\{q_{mn'} : n' \in \mathcal{N}(m)\backslash n\}$. The algorithm would produce the exact posterior probabilities of all the bits if the bipartite graph defined by the matrix $\mathbf{H}$ contained no cycles [10].

Initialization. The variables q_{mn}^0 and q_{mn}^1 are initialized to the values f_n^0 and f_n^1 respectively.

Horizontal step. We define $\delta q_{mn} \equiv q_{mn}^0 - q_{mn}^1$ and compute for each m, n:

$$\delta r_{mn} = \prod_{n' \in \mathcal{N}(m)\backslash n} \delta q_{mn'} \tag{1}$$

then set $r_{mn}^0 = \frac{1}{2}(1 + \delta r_{mn})$ and $r_{mn}^1 = \frac{1}{2}(1 - \delta r_{mn})$.

Vertical step. For each n and m and for $x = 0, 1$ we update:

$$q_{mn}^x = \alpha_{mn}\, f_n^x \prod_{m' \in \mathcal{M}(n)\backslash m} r_{m'n}^x \tag{2}$$

where α_{mn} is chosen such that $q_{mn}^0 + q_{mn}^1 = 1$. We can also update the 'pseudoposterior probabilities' q_n^0 and q_n^1, given by:

$$q_n^x = \alpha_n\, f_n^x \prod_{m \in \mathcal{M}(n)} r_{mn}^x. \tag{3}$$

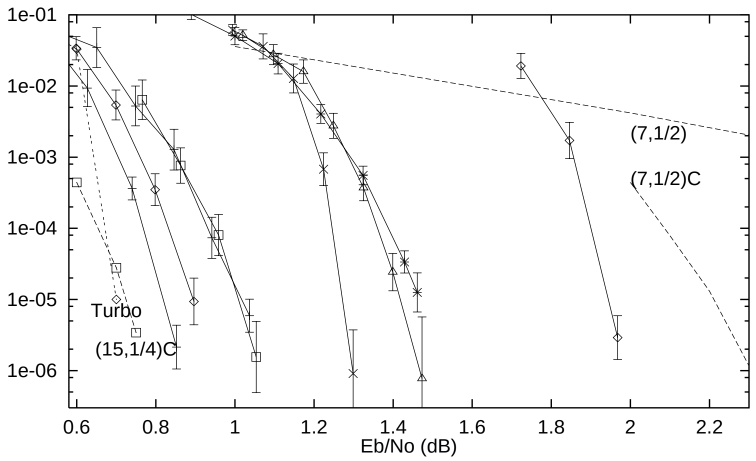

Figure 1: GL codes' performance over Gaussian channel (solid curves) compared with that of standard textbook codes (dotted curves at right) and state-of-the-art codes (dotted curves at left).

These quantities are used to create a tentative bit-by-bit decoding $\hat{\mathbf{x}}$; if $\mathbf{H}\hat{\mathbf{x}} = 0$ then the decoding algorithm halts. Otherwise, the algorithm repeats from the horizontal step. A failure is declared if some maximum number of iterations (*e.g.*, 100) occurs without a valid decoding.

Results. Figure 1 compares the performance of GL codes with textbook codes and with state of the art codes. The vertical axis shows the empirical bit error probability. **Textbook codes:** The curve labelled (7,1/2) shows the performance of a rate $\frac{1}{2}$ convolutional code with constraint length 7, known as the de facto standard for satellite communications [7]. The curve (7,1/2)C shows the performance of the concatenated code composed of the same convolutional code and a Reed-Solomon code. **State of the art:** The curve (15,1/4)C shows the performance of an extremely expensive and computer intensive concatenated code developed at JPL based on a constraint length 15, rate $\frac{1}{4}$ convolutional code (data courtesy of R.J. McEliece.) The curve labelled Turbo shows the performance of the rate $\frac{1}{2}$ Turbo code described in [2]. (Better Turbo codes have since been reported [3].) **GL codes:** From left to right the codes had the following parameters (N, K, R): $(29507, 9507, 0.322)$ (construction 2B); $(15000, 5000, 0.333)$ (2A); $(14971, 4971, 0.332)$ (2B); $(65389, 32621, 0.499)$ (1B); $(19839, 9839, 0.496)$ (1B); $(13298, 3296, 0.248)$ (1B); $(29331, 19331, 0.659)$ (1B). It should be emphasised that *all* the errors made by the GL codes were *detected* errors: the decoding algorithm reported the fact that it had failed.

Our results show that performance substantially better than that of standard convolutional and concatenated codes can be achieved; indeed the performance is almost as close to the Shannon limit as that of Turbo codes [2]. It seems that the best results are obtained by making the weight per column as small as possible (construction 2A). Unsurprisingly, codes with larger block length are better. In terms of the value of E_b/N_0, the best codes were ones with rates between $\frac{1}{2}$ and $\frac{1}{3}$.

Cost. In a brute force approach, the time to create the code scales as N^3, where N is the block size. The encoding time scales as N^2, but encoding involves only binary arithmetic, so for the block lengths studied here it takes considerably less time than the simulation of the Gaussian channel. It may be possible to reduce encoding time using sparse matrix techniques. Decoding involves approximately $6Nt$ floating point multiplies per iteration, so the total number of operations per decoded bit (assuming 20 iterations) is about $120t/R$, independent of block length. For the codes presented here, this is about 800 operations.

This work not only confirms the assertion [1] that 'we can think of good codes, and even decode them', but also that 'we could think of them, and decode them, thirty-five years ago'.

Acknowledgements

DJCM is grateful to Robert McEliece and Roger Sewell for helpful discussions.

References

[1] G. Battail. We can think of good codes, and even decode them. In P. Camion, P. Charpin, and S. Harari, editors, *Eurocode '92. Udine, Italy, 26-30 October*, number 339 in CISM Courses and Lectures, pages 353–368. Springer, Wien, 1993.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo-codes. In *Proc. 1993 IEEE International Conference on Communications, Geneva, Switzerland*, pages 1064–1070, 1993.

[3] D. Divsilar and F. Pollara. On the design of turbo codes. Technical Report TDA 42-123, Jet Propulsion Laboratory, Pasadena, November 1995.

[4] G. D. Forney. Concatenated codes. Technical Report 37, MIT, 1966.

[5] R. G. Gallager. Low density parity check codes. *IRE Trans. Info. Theory*, IT-8:21–28, Jan 1962.

[6] R. G. Gallager. *Low Density Parity Check Codes.* Number 21 in Research monograph series. MIT Press, Cambridge, Mass., 1963.

[7] S. W. Golomb, R. E. Peile, and R. A. Scholtz. *Basic Concepts in Information Theory and Coding: The Adventures of Secret Agent 00111.* Plenum Press, New York, 1994.

[8] D. J. C. MacKay and R. M. Neal. Good codes based on very sparse matrices. In Colin Boyd, editor, *Cryptography and Coding. 5th IMA Conference*, number 1025 in Lecture Notes in Computer Science, pages 100–111. Springer, Berlin, 1995.

[9] D. J. C. MacKay and R. M. Neal. Good error correcting codes based on very sparse matrices. Available from `http://wol.ra.phy.cam.ac.uk/`, 1996.

[10] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, 1988.

Version 2.0