Electronic Notes in Theoretical Computer Science 74 (2003)
URL: `http://www.elsevier.nl/locate/entcs/volume74.html` 8 pages

Performance of Low Density Parity Check Codes as a Function of Actual and Assumed Noise Levels

David J.C. MacKay [1] and Christopher P. Hesketh

Cavendish Laboratory,
Cambridge, CB3 0HE, United Kingdom.

Abstract

We investigate how sensitive Gallager's codes are, when decoded by the sum-product algorithm, to the assumed noise level. We have found a remarkably simple function that fits the empirical results as a function of the actual noise level at both high and low noise levels.

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 $\mathbf{Hx} = 0$ for all codewords $\mathbf{x}$. In 1962, Gallager reported work on binary codes defined in terms of low density parity check matrices [2,3]. The matrix $\mathbf{H}$ is defined in a non-systematic form; each column of $\mathbf{H}$ has the same small predetermined weight (*e.g.*, 3) and the weight per row is also uniform; the matrix $\mathbf{H}$ is constructed at random subject to these constraints. When these codes are decoded using Gallager's iterative probabilistic decoding method, also known as the sum-product algorithm or belief propagation, their empirical performance is found to be excellent [4,5].

In this paper, we investigate the dependence of the performance of a low density parity check code on both the assumed and actual noise levels of a binary symmetric channel and a Gaussian channel. All the experiments make use of binary codes whose parity check matrices are randomly generated with weight per column $t = 3$ using construction methods 1A and 1B of [4].

The present work has four motivations. First, since the decoding algorithm (reviewed in detail in [4,5]) is somewhat *ad hoc*, it seems an interesting experiment to 'tweak' it, checking to see whether any of its control parameters can be set to better values than their standard values; specifically, we tweak the assumed noise level to see if setting it to higher or lower values improves the performance of the sum-product algorithm. Second, this project sheds

[1] Email: `mackay@mrao.cam.ac.uk`

light on the nature of the errors made by low density parity check codes. Are decoding errors associated solely with unusually high noise levels, or are there particular noise patterns which are associated with decoding errors? Third, in practical situations, we are unlikely to know the channel's characteristics exactly, and it will be helpful if our decoder is robust to mismatches between the assumed channel model and the true channel statistics. Fourth, the measurements we make here give an indication of how much computer time can be saved, when simulating these codes, by using biased sampling methods. An unexpected spin-off of this work is an empirical formula which accurately describes the iterative decoder's error probability as a function of signal to noise ratio at both high and low noise levels.

Method

Our first experiments used a small low density parity check code with rate 1/2 and block length $N = 2000$ over a binary symmetric channel that flips a fraction n of the bits. We will call n the 'actual crossover probability'. We used the probability-based representation for belief propagation decoding described in [5]. [Results in another representation such as the log-likelihood representation would have been identical, apart from differences produced by rounding error.] In each decoding run we ran the decoder until it either found a valid codeword or timed out after a maximum of 200 iterations. If the maximum number of iterations was reached then the decoding was considered a failure. In all the simulations reported in this paper, whenever a valid codeword was found, it was the correct one. (Unlike turbo codes [1], low density parity check codes are never found to make undetected errors, because they have good distance properties [4,5].) The noise vectors were generated to have weight exactly nN, where n was the 'actual noise level', which we varied from experiment to experiment (note that we did *not* draw the noise vector from a Bernoulli distribution, as would be appropriate if we were discussing the binary symmetric channel with independent identically distributed noise; the results for this case can be obtained by convolving the results we find here, as a function of n, with the probability of the flipped fraction n given the flip probability). The decoder assumed a crossover probability of f.

We performed two types of experiments. First, with the actual noise level n fixed, we varied the assumed noise level f and observed the failure probability. Second, we varied the actual noise level n while the assumed noise level remained fixed at f. The results of these two experiments are shown in figure 1.

Interestingly, neither an increase nor a decrease in the *assumed* noise level gives significantly better performance than does leaving it set to the actual noise level. The block error probability is not a very sensitive function of the assumed noise level. To the left of the optimum, the graph of log(block error probability) versus $\log(f)$ appears to have slope about -1, indicating that

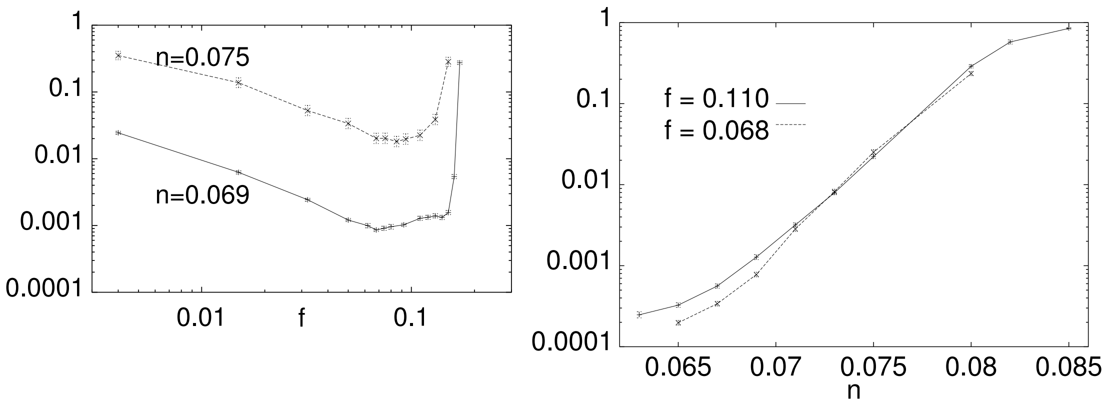

Fig. 1. Low density parity check code's block error probability (a) as a function of assumed noise level when the actual noise level was fixed; (b) as a function of actual noise level for fixed assumed noise level. These results are for a code of block length $N = 2000$ and a binary symmetric channel. The error bars show approximate 95% confidence intervals.

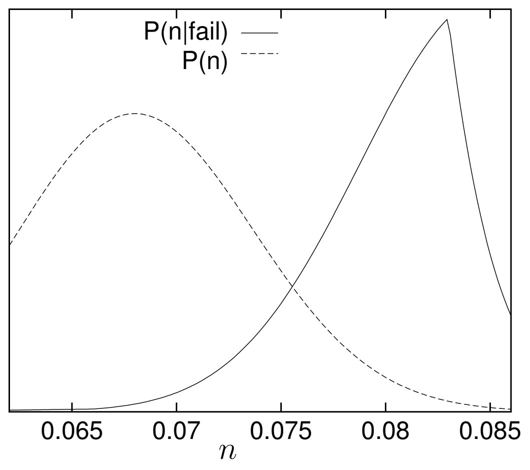

Fig. 2. Probability distribution, $P(n|\text{fail})$, of weight of noise vectors that cause failures given that the distribution of noise vector weights $P(n)$ is binomial with mean $\bar{n} = 0.068$, and assuming $f = 0.068$. A piecewise linear approximation to the graph of $\log P(\text{fail}|n)$ given in figure 1(b) was used to deduce the posterior probability $P(n|\text{fail})$. These results are for a code of block length $N = 2000$ and a binary symmetric channel.

if f is too small by a factor of two, we suffer a doubling of the block error probability. To the right of the minimum the slope appears to be even smaller, so that a doubling of the assumed f increases the block error probability by less than a factor of two. When the assumed noise level is far too large (well beyond the Shannon limit of 0.11 for a rate 1/2 code) there is a catastrophic increase in the error probability. This corresponds to the decoding algorithm's always settling down in a non-committal state.

In the second plot, the results for fixed f as a function of the actual noise

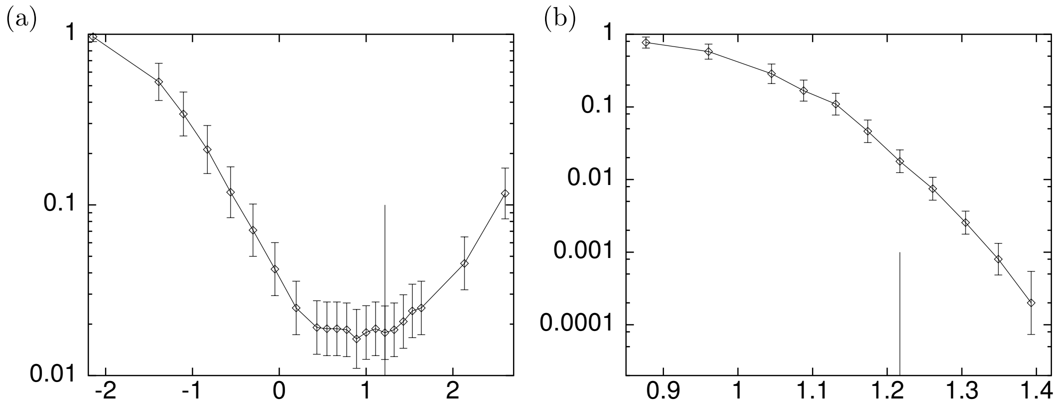

Fig. 3. Low density parity check code's block error probability for a Gaussian channel (a) as a function of assumed signal to noise ratio when the actual s.n.r. was fixed; (b) as a function of actual s.n.r. for fixed assumed s.n.r.. Horizontal axis shows signal to noise ratio E_b/N_0 in decibels. Vertical axis shows observed block error probability, with error bars showing approximate 95% confidence intervals. The vertical line in (a) indicates the fixed actual s.n.r. (1.22dB), and in (b) the vertical line indicates the fixed assumed s.n.r. (1.22dB). These results are for a code of block length $N = 13298$.

level n show a close to linear relationship between log(block error probability) and n in the low noise regime.

From this curve we can deduce the probability distribution of the weight of the error-causing noise vectors when the distribution of noise vectors is binomial with any density of interest, for example $\bar{n} = 0.068$, and with the assumed $f = 0.068$. The prior probability of the weight $w \equiv nN$ is

$$(1) \quad P(w) = \binom{N}{w} \bar{n}^w (1-\bar{n})^{(N-w)}$$

$$(2) \qquad \simeq \exp\left[N\left(H_2(n) + n\log\bar{n} + (1-n)\log(1-\bar{n})\right)\right]$$

where $H_2(n) = -n\log n + (1-n)\log(1-n)$. (All logs are natural logarithms.) The probability of failure $P(\text{fail}|n)$ is given in figure 1(b). The posterior probability of w (equivalently, of n) given that there is a failure, is given by

$$(3) \qquad P(n|\text{fail}) \propto P(\text{fail}|n)P(n)$$

and is shown, along with $P(n)$, in figure 2. This graph indicates that (at least for block lengths about 2000) decoding errors are almost all associated with noise vectors of above-average weight.

Results for a longer blocklength code on a Gaussian channel

Just as in the case of the binary symmetric channel where we distinguished between the mean error rate of the channel and the actual fraction of bits flipped, when studying an additive white Gaussian noise channel we distinguish between the 'mean' signal to noise level (s.n.r.) and the 'actual' s.n.r.. In this paper we control the 'actual' s.n.r.. Results for a code of block length

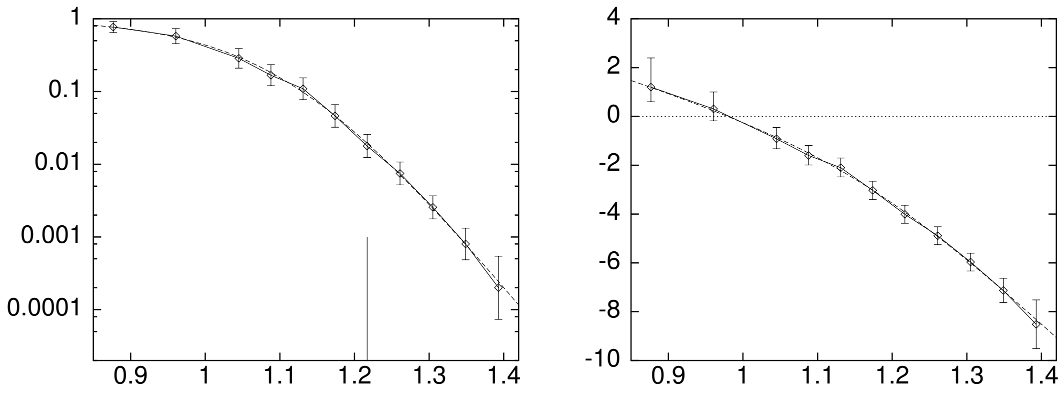

Fig. 4. Empirical fitting: Low density parity check code's block error probability for a Gaussian channel (solid line) as a function of actual s.n.r. for fixed assumed s.n.r.. The dashed line shows the fitted function $f(x) = 1/(1 + a^{c(x-b)})^{1/\log(a)}$ with $a = 1.42$, $b = 1.109$ and $c = 28.7$. These results are for a code of block length $N = 13298$. The lefthand figure shows p_B on a logarithmic scale, and the right hand figure shows p_B on a logit scale, $\log[p_B/(1-p_B)]$. The error bars show approximate 95% confidence intervals.

$N = 13298$ are shown in figure 3. The horizontal axis shows the s.n.r. E_b/N_0. The broad features of the curves are similar to those found in the case of the binary symmetric channel. The decoder's performance is insensitive to an assumed value of E_b/N_0 that is erroneously small by up to 1dB, but it *is* quite sensitive to the assumed value of E_b/N_0 if it is too large. If the assumed value of E_b/N_0 is too large by a factor of 1.4, the error probability increases by a factor of 10.

Empirical model

We found that the results shown in figure 3(b) can be fitted remarkably accurately by a simple function

$$f(x) = 1/(1 + a^{c(x-b)})^{1/\log(a)}, \tag{4}$$

where for this code the fitted parameters are found using C. Grammes's `gnufit` to be $a = 1.42 \pm 0.01$, $b = 1.109 \pm 0.003$ and $c = 28.7 \pm 0.3$. This function, shown in figure 4, can be thought of as a 'soft min' of the two functions $f_1(x) = 1$ and $f_2(x) = e^{-c(x-b)}$. We can also write this function as

$$g(x) = \left[\frac{1}{1 + \exp(c'(x-b))}\right]^{a'} \tag{5}$$

where $a' = 2.86 \pm 0.06$, $b = 1.109 \pm 0.003$ $c' = 10.0 \pm 0.1$; this function can be thought of as the probability of a 'noisy and' of a' probabilities, each varying as $1/(1 + \exp(c'(x-b)))$. This is a pleasingly compact description of the error probability which may shed light on the decoding algorithm's failures. If we accept this approximation, we can easily obtain the standard graphs of error probability versus mean signal to noise ratio by a simple integration with respect to the actual noise level, whose probability density is a chi-squared distribution. This empirical formula may thus allow a saving in simulations, since the three parameters may be fitted using data from the cheap-to-simulate 'waterfall' portion of the curve.

Is there any simple structure in the ensemble of noise vectors that cause decoding errors?

Among the set of noise vectors with some given weight, some cause decoding errors and some do not. It would be nice to understand what it is about the error-causing noise vectors that makes them problematic. We might hope that simple statistics such as the mean error-causing noise vector might show significant differences from the uniform mean noise vector, *i.e.*, that the code is more susceptible to bit errors at particular bit positions that others (although in the case of convolutional codes, this is not the case; the first order statistics contain no information). To investigate this idea, we counted how many times r_n each of the 13,298 bits was in error in a sample of one hundred error-causing noise vectors and tested these 13,298 numbers $\{r_n\}$ against the null hypothesis, that all bits are equally likely to be 1 in an error-causing noise vector. [Here, n is an integer running over the transmitted bits $n \in \{1, \ldots, N\}$.]

Results for the Gaussian channel are shown in figure 5. Let r_n denote the number of times bit n is equal to one in the 100 error-causing noise vectors. The figure shows a histogram of the values of the numbers numbers $r_1 \ldots r_N$. Under the null hypothesis we expect the histogram of r to be generated from a binomial distribution. If, however, there are some noise bits which are more likely to be 1 in error-causing noise vectors then we expect this histogram to have a broader shape than a binomial distribution, with larger values and smaller values of r both occurring more frequently than predicted by the null hypothesis. The histograms in figure 5 show no such effects: a chi-squared test, described below, gave $\chi^2 = 23.7$ where the number of degrees of freedom (which is the expectation of χ^2) was $\nu = 25$.

The chi-squared test.

We grouped the bins in the histogram together into new bins such that the expected count c_n in any new bin was greater than 5 (this being the condition for the test we used to be valid (B.D. Ripley, personal communication)); we then computed $\chi^2 \equiv \sum_n (r_n - c_n)^2/\text{var}_n$, where $\text{var}_n = c_n(1 - c_n/N)$; the expectation of χ^2, ν, is the number of new bins minus one.

Conclusion

The sum-product decoder for low density parity check codes appears to be robust to incorrect assumptions about the noise level, if one errs towards assuming a larger noise level than actually exists; if one assumes too small a noise level, there may be an appreciable deterioration in performance.

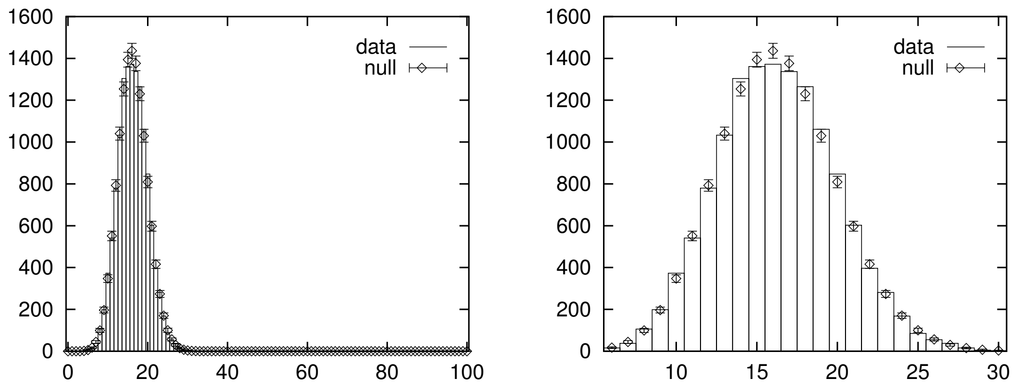

Fig. 5. Statistics of the empirical mean noise vector compared with those predicted by the null hypothesis. These results are for a code of block length $N = 13298$ and a binary symmetric channel, based on 100 block errors. The noise level was adjusted to give a block error rate of about 10^{-2}. The figures compare the histogram of the numbers $\{r_n\}$ defined in the text with the predictions of the null hypothesis. The right figure shows detail from the left figure.

We have not detected any significant first order pattern in the noise vectors that cause decoding errors. The nature of these error-causing noise vectors remains an interesting topic for further research.

We have found a remarkably compact formula that describes the error probability as a function of noise level in the case of a Gaussian channel, giving a good fit in both the high probability of error and low probability of error regimes.

Acknowledgements

DJCM thanks G.D. Forney for helpful discussions. DJCM was supported in part by the Royal Society Smithson research fellowship and by a grant from the Gatsby charitable foundation. CPH thanks Cambridge central computing service and Ian Farrer for generous provision of computing resources, and Andy Lupini for helpful discussions.

References

[1] 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.

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

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

[4] D. J. C. MacKay. Good error correcting codes based on very sparse matrices. *IEEE Transactions on Information Theory*, 45(2):399–431, 1999.

[5] D. J. C. MacKay and R. M. Neal. Near Shannon limit performance of low density parity check codes. *Electronics Letters*, 32(18):1645–1646, August 1996. Reprinted *Electronics Letters*, **33**(6):457–458, March 1997.