A Free Energy Minimization Framework for Inference Problems in Modulo 2 Arithmetic

David J.C. MacKay*

Cavendish Laboratory
Cambridge CB3 0HE
United Kindom

mackay@mrao.cam.ac.uk

Abstract. This paper studies the task of inferring a binary vector $\mathbf{s}$ given noisy observations of the binary vector $\mathbf{t} = \mathbf{A}\mathbf{s}$ modulo 2, where $\mathbf{A}$ is an $M \times N$ binary matrix. This task arises in correlation attack on a class of stream ciphers and in the decoding of error correcting codes.

The unknown binary vector is replaced by a real vector of probabilities that are optimized by variational free energy minimization. The derived algorithms converge in computational time of order between w_A and Nw_A, where w_A is the number of 1s in the matrix $\mathbf{A}$, but convergence to the correct solution is not guaranteed.

Applied to error correcting codes based on sparse matrices $\mathbf{A}$, these algorithms give a system with empirical performance comparable to that of BCH and Reed-Muller codes.

Applied to the inference of the state of a linear feedback shift register given the noisy output sequence, the algorithms offer a principled version of Meier and Staffelbach's (1989) algorithm B, thereby resolving the open problem posed at the end of their paper. The algorithms presented here appear to give superior performance.

1 The problem addressed in this paper

Consider three binary vectors: $\mathbf{s}$ with components s_n, $n = 1 \ldots N$, and $\mathbf{r}$ and $\mathbf{n}$ with components r_m, n_m, $m = 1 \ldots M$, related by:

$$(\mathbf{A}\mathbf{s} + \mathbf{n}) \bmod 2 = \mathbf{r} \tag{1}$$

where $\mathbf{A}$ is a binary matrix. Our task is to infer $\mathbf{s}$ given $\mathbf{r}$ and $\mathbf{A}$, and given assumptions about the statistical properties of $\mathbf{s}$ and $\mathbf{n}$.

This problem arises, for example, in the decoding of a noisy signal transmitted using a linear code $\mathbf{A}$. As a simple illustration, the (7,4) Hamming code takes $N = 4$ signal bits, $\mathbf{s}$, and transmits them followed by three parity check bits.

* Supported by the Royal Society Smithson Research Fellowship

The $M = 7$ transmitted symbols are given by $\mathbf{t} = \mathbf{A}\mathbf{s} \bmod 2$, where

$$\mathbf{A} = \begin{bmatrix} 1\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \\ 1\,0\,1\,1 \\ 1\,1\,0\,1 \\ 1\,1\,1\,0 \end{bmatrix}$$

The noise vector $\mathbf{n}$ describes the corruption of these bits by the communication channel. The received message is $\mathbf{r} = \mathbf{t} + \mathbf{n} \bmod 2$. The receiver's task is to infer $\mathbf{s}$ given $\mathbf{r}$ and the assumed noise properties of the channel. For the Hamming code above, this is not a difficult task, but as N and M become large and as the number of 1s in the matrix $\mathbf{A}$ increases, the inference of $\mathbf{s}$ in a time less than exponential in N becomes more challenging, for general $\mathbf{A}$. Indeed, the general decoding problem for linear codes is NP-complete (Berlekamp, McEliece and van Tilborg 1978).

The problem defined in equation (1) also arises in the inference of the sequence of a linear feedback shift register (LFSR) from noisy observations (Meier and Staffelbach 1989, Mihaljević and Golić 1992, Mihaljević and Golić 1993, Anderson 1995).

This paper presents a fast algorithm for attempting to solve these tasks.

Assumptions

I assume that the prior probability distribution of $\mathbf{s}$ and $\mathbf{n}$ is separable, *i.e.*, $P(\mathbf{s}, \mathbf{n}) = P(\mathbf{s})P(\mathbf{n}) = \prod_n P(s_n) \prod_m P(n_m)$. Defining the transmission $\mathbf{t} = \mathbf{A}\mathbf{s} \bmod 2$, the probability of the observed data $\mathbf{r}$ as a function of $\mathbf{s}$ (the 'likelihood function') can be written:

$$P(\mathbf{r}|\mathbf{s}, \mathbf{A}) = \prod_m e_m^{t_m(\mathbf{s})}(1 - e_m)^{(1-t_m(\mathbf{s}))},$$

where $\mathbf{e}$ is a vector of probabilities indexed by m given by $e_m = P(n_m = 1)$ if $z_m = 0$ and $e_m = P(n_m = 0)$ if $z_m = 1$. This likelihood function is fundamental to any probabilistic approach. The log likelihood can be written:

$$\log P(\mathbf{r}|\mathbf{s}, \mathbf{A}) = \sum_m t_m(\mathbf{s}) \log \frac{e_m}{1 - e_m} + \text{const.} \tag{2}$$

$$\equiv \sum_m t_m(\mathbf{s})\, g_m + \text{const.} \tag{3}$$

where $g_m \equiv \log[e_m/(1 - e_m)]$. Thus the natural norm for measuring the distance between $\mathbf{t}$ and $\mathbf{e}$ is $\sum_m t_m g_m(e_m)$.

The task is to infer $\mathbf{s}$ given $\mathbf{A}$ and the data $\mathbf{r}$. The posterior distribution of $\mathbf{s}$ is, by Bayes's theorem:

$$P(\mathbf{s}|\mathbf{r}, \mathbf{A}) = \frac{P(\mathbf{r}|\mathbf{s}, \mathbf{A})P(\mathbf{s})}{P(\mathbf{r}|\mathbf{A})}. \tag{4}$$

I assume our aim is to find the most probable $\mathbf{s}$, *i.e.*, the $\mathbf{s}$ that maximizes the expression (4) over $\mathbf{s}$. I assume however that an exhaustive search over the 2^N possible sequences $\mathbf{s}$ is not permitted. One way to attack a discrete combinatorial problem is to create a related problem in which the discrete variables $\mathbf{s}$ are replaced by real variables, over which a continuous optimization can then be performed [see for example (Hopfield and Tank 1985, Aiyer, Niranjan and Fallside 1990, Durbin and Willshaw 1987, Peterson and Soderberg 1989, Gee and Prager 1994, Blake and Zisserman 1987)]. In the present context, the question then is 'how should one generalize the posterior probability (4) to the case where $\mathbf{s}$ is replaced by a vector with real components?' An appealing way to answer this question is to derive the continuous representation in terms of an approximation to probabilistic inference.

2 Free Energy Minimization

The variational free energy minimization method (Feynman 1972) takes an 'awkward' probability distribution such as the one in (4), and attempts to approximate it by a simpler distribution $Q(\mathbf{s};\theta)$, parameterized by a vector of parameters θ. For brevity in the following general description I will denote the complex probability distribution $P(\mathbf{s}|\mathbf{A},\mathbf{r})$ by $P(\mathbf{s})$. The measure of quality of the approximating distribution is the variational free energy,

$$F(\theta) = \sum_{\mathbf{s}} Q(\mathbf{s};\theta) \log \frac{Q(\mathbf{s};\theta)}{P(\mathbf{s})}.$$

The function $F(\theta)$ has a lower bound of zero which is attained only by a θ such that $Q(\mathbf{s};\theta) = P(\mathbf{s})$ for all $\mathbf{s}$. Alternatively, if $P(\mathbf{s})$ is not normalized, and we define $Z = \sum_{\mathbf{s}} P(\mathbf{s})$, then F has a lower bound of $-\log Z$, attained only by θ such that $Q = P/Z$. The variational method used in this paper is traditionally used in statistical Physics to estimate $\log Z$, but here, $\log Z$ is just an additive constant which we ignore.

When Q has a sufficiently simple form, the optimization of F over θ may be a tractable problem, even though F is defined by a sum over all values of $\mathbf{s}$. We find a θ^* that minimizes $F(\theta)$ in the hope that the approximating distribution $Q(\mathbf{s};\theta^*)$ will give useful information about $P(\mathbf{s})$. Specifically, we might hope that the $\mathbf{s}$ that maximizes $Q(\mathbf{s};\theta^*)$ is a good guess for the $\mathbf{s}$ that maximizes $P(\mathbf{s})$. Generically, free energy minimization produces approximating distributions $Q(\mathbf{s};\theta^*)$ that are more compact than the true distribution $P(\mathbf{s})$.

3 Free Energy Minimization for Equation (4)

I take Q to be a separable distribution,

$$Q(\mathbf{s};\theta) \equiv \prod_n q_n(s_n;\theta_n)$$

with θ_n defining the probabilities q_n thus:

$$q_n(s_n=1;\theta_n) = \frac{1}{1+e^{-\theta_n}} \equiv q_n^1; \;\; q_n(s_n=0;\theta_n) = \frac{1}{1+e^{\theta_n}} \equiv q_n^0.$$

This is a nice parameterization because the log probability ratio is $\theta_n = \log(q_n^1/q_n^0)$. The variational free energy is defined to be:

$$F(\theta) = \sum_{\mathbf{s}} Q(\mathbf{s};\theta) \log \frac{Q(\mathbf{s};\theta)}{P(\mathbf{r}|\mathbf{s},\mathbf{A})P(\mathbf{s})}.$$

I now derive an algorithm for computing F and its gradient with respect to θ in a time that is proportional to the weight of $\mathbf{A}$, w_A (*i.e.*, the number of ones in $\mathbf{A}$). The free energy separates into three terms, $F(\theta) = E_L(\theta) + E_P(\theta) - S(\theta)$, where the 'entropy' is:

$$S(\theta) \equiv - \sum_{\mathbf{s}} Q(\mathbf{s};\theta) \log Q(\mathbf{s};\theta) = - \sum_n \left[q_n^0 \log q_n^0 + q_n^1 \log q_n^1 \right],$$

with derivative: $\frac{\partial}{\partial \theta_n} S(\theta) = -q_n^0 q_n^1 \theta_n$; the 'prior energy' is:

$$E_P(\theta) \equiv - \sum_{\mathbf{s}} Q(\mathbf{s};\theta) \log P(\mathbf{s}) = - \sum_n b_n q_n^1$$

where $b_n = \log[P(s_n\!=\!1)/P(s_n\!=\!0)]$, and has derivative $\frac{\partial}{\partial \theta_n} E_P(\theta) = -q_n^0 q_n^1 b_n$; and the 'likelihood energy' is:

$$E_L(\theta) \equiv - \sum_{\mathbf{s}} Q(\mathbf{s};\theta) \log P(\mathbf{r}|\mathbf{s},\mathbf{A}) = - \sum_m g_m \sum_{\mathbf{s}} Q(\mathbf{s};\theta)\, t_m(\mathbf{s}) + \text{const}.$$

The additive constant is independent of θ and will now be omitted. To evaluate E_L, we need the average value of $t_m(\mathbf{s})$ under the separable distribution $Q(\mathbf{s};\theta)$, that is, the probability that $\sum_n A_{mn} s_n \bmod 2 = 1$ under that distribution.

Forward algorithm. We can compute this probability for each m by a recursion involving a sequence of probabilities $p^1_{m,\nu}$ and $p^0_{m,\nu}$ for $\nu = 1 \ldots N$. These are defined to be the probabilities that the partial sum $t_m^{1\nu} = \sum_{n=1}^{\nu} A_{mn} s_n \bmod 2$ is equal to 1 and 0 respectively. These probabilities satisfy:

$$\left.\begin{aligned} p^1_{m,\nu} &= q^0_\nu p^1_{m,\nu-1} + q^1_\nu p^0_{m,\nu-1} \\ p^0_{m,\nu} &= q^0_\nu p^0_{m,\nu-1} + q^1_\nu p^1_{m,\nu-1} \end{aligned}\right\} \text{ if } A_{m\nu} = 1;$$
$$\left.\begin{aligned} p^1_{m,\nu} &= p^1_{m,\nu-1} \\ p^0_{m,\nu} &= p^0_{m,\nu-1} \end{aligned}\right\} \text{ if } A_{m\nu} = 0. \tag{5}$$

The initial condition is $p^1_{m,0} = 0, p^0_{m,0} = 1$. The desired quantity is obtained in a time that is linear in the weight of row m of $\mathbf{A}$:

$$\sum_{\mathbf{s}} Q(\mathbf{s};\theta)\, t_m(\mathbf{s}) = p^1_{m,N}.$$

The quantity $p^1_{m,N}$ is a generalization to a continuous space of the product $t_m = \mathbf{A}_m \mathbf{s} \bmod 2$, with the satisfying property that if any one of the contributing terms q^1_ν is equal to 0.5, then the resulting value of $p^1_{m,N}$ is 0.5.

The energy E_L is then given by:

$$E_L(\theta) = -\sum_m g_m p^1_{m,N}.$$

Reverse algorithm. To obtain the derivative of the energy E_L with respect to θ_n it is necessary to perform the same number of computations again. We introduce another 'reverse' sequence of probabilities $r^1_{m,\nu}$ and $r^0_{m,\nu}$ for $\nu = N \ldots 1$, defined to be the probabilities that the partial sum $t^{\nu N}_m = \sum_{n=\nu}^N A_{mn} s_n \bmod 2$ is equal to 1 and 0 respectively. These can be computed by a recursive procedure equivalent to that in equation (5). Now note that $p^1_{m,N}$ can be written thus, for any n:

$$p^1_{m,N} = q^0_n \left(p^1_{m,n-1} r^0_{m,n+1} + p^0_{m,n-1} r^1_{m,n+1}\right) + q^1_n \left(p^1_{m,n-1} r^1_{m,n+1} + p^0_{m,n-1} r^0_{m,n+1}\right).$$

Having pulled out the θ_n dependence (in the two factors q^0_n and q^1_n), it is easy to obtain the derivative:

$$\frac{\partial}{\partial \theta_n} E_L(\theta) = -q^0_n q^1_n \sum_m g_m d_{mn}$$

where $d_{mn} = \left(p^1_{m,n-1} r^1_{m,n+1} + p^0_{m,n-1} r^0_{m,n+1}\right) - \left(p^1_{m,n-1} r^0_{m,n+1} + p^0_{m,n-1} r^1_{m,n+1}\right)$.

Total derivative

The derivative of the free energy is:

$$\frac{\partial F}{\partial \theta_n} = q^0_n q^1_n \left[\theta_n - b_n - \sum_m g_m d_{mn}\right]. \tag{6}$$

Optimizers

This derivative can be inserted into a variety of continuous optimizers. I have implemented both conjugate gradient optimizers and 'reestimation' optimizers and found the latter, which I now describe, to be superior. The reestimation method is motivated by the form of the derivative (6); setting it to zero, we obtain the iterative prescription:

$$\theta_n := b_n + \sum_m g_m d_{mn},$$

which I call the reestimation optimizer. It can be implemented with either *synchronous* or *sequential* dynamics, that is, we can update all θ_n simultaneously, or one at a time. The sequential reestimation optimizer is guaranteed to reduce the free energy on every step, because everything to the right of θ_n in equation (6) is independent of θ_n. The sequential optimizer can be efficiently interleaved with the reverse recursion; the reverse probability r^1_ν is evaluated from $r^1_{\nu+1}$ for each m after θ_ν has been updated. The synchronous optimizer does not have an associated Lyapunov function, so it is not guaranteed to be a stable optimizer, but empirically it sometimes performs better.

Optimizers of the free energy can be modified by introducing 'annealing' or 'graduated non-convexity' techniques (Blake and Zisserman 1987, Van den Bout and Miller 1989), in which the non-convexity of the objective function F is switched on gradually by varying an inverse temperature parameter β from 0 to 1. This annealing procedure is intended to prevent the algorithm running headlong into the minimum that the initial gradient points towards. We define:

$$F(\theta, \beta) = \beta E_L(\theta) + \beta_P E_P(\theta) - S(\theta), \tag{7}$$

and perform a sequence of minimizations of this function with successively larger values of β, each starting where the previous one stopped. If we choose $\beta_P \equiv \beta$ then β influences both the likelihood energy and the prior energy, and if $\beta_P \equiv 1$ then β influences just the likelihood energy. The gradient of $F(\theta, \beta)$ with respect to θ is identical to the gradient of F in equation (6) except that the energy terms are multiplied by β_P and β. This annealing procedure is deterministic and does not involve any simulated noise.

Comments

None of these algorithms is expected to work in all cases, because a) there may be multiple free energy optima; b) the globally optimal $\mathbf{s}$ might not be associated with any of these free energy optima; c) the task is a probabilistic task, so even an exhaustive search is not guaranteed always to identify the correct vector $\mathbf{s}$.

Any particular problem can be reparameterized into other representations $\mathbf{s}' = \mathbf{s}\mathbf{U}$ with new matrices $\mathbf{A}' = \mathbf{U}^{-1}\mathbf{A}$. The success of the algorithms is expected to depend crucially on the choice of representation. The algorithms are expected to work best if $\mathbf{A}$ is sparse and the true posterior distribution over $\mathbf{s}$ is close to separable.

Computational complexity

The gradient of the free energy can be calculated in time linear in w_A. The algorithms are expected to take of order 1, or at most N, gradient evaluations to converge, so that the total time taken is of order between w_A and $w_A N$.

The space requirements of the sequential reestimation optimizer are the most demanding (but not severe): memory proportional to w_A is required.

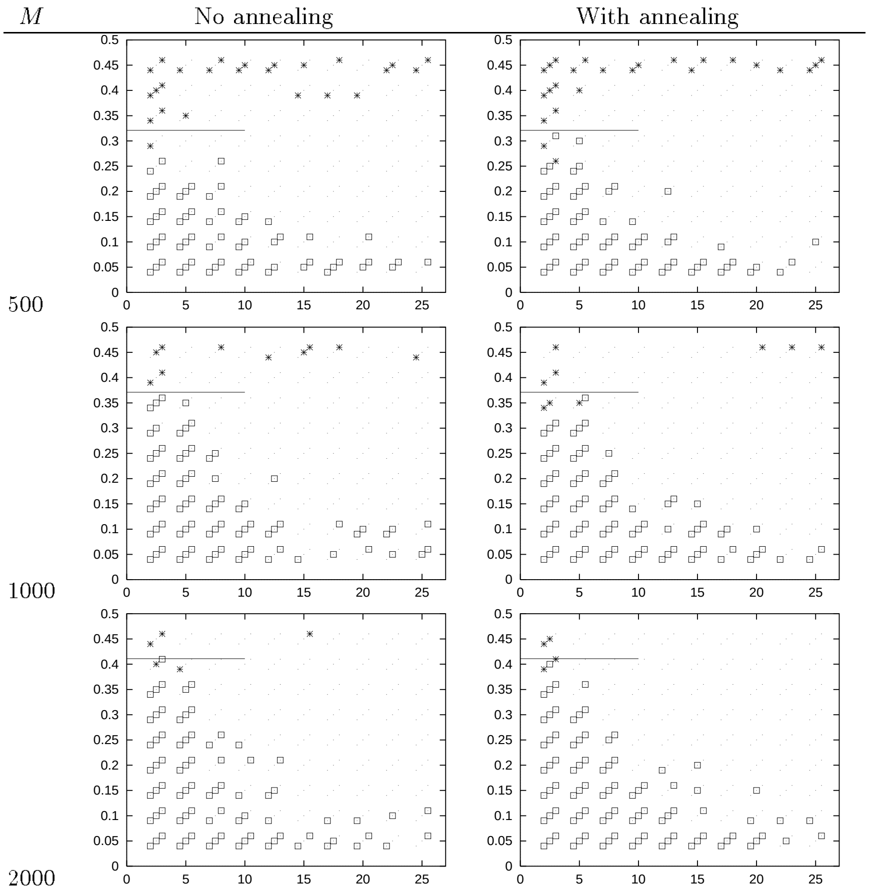

Fig. 1. Performance of synchronous reestimation algorithm without and with annealing: N=50

Horizontal axis: average number of 1s per row of $\mathbf{A}$. Vertical axis: noise level f_n. Three experiments were conducted at a grid of values. Outcome is represented by: box = 'correct'; star = 'ok'; dot = 'wrong'. Two points in each triplet have been displaced so that they do not overlap. The horizontal line on each graph indicates an information theoretic bound on the noise level beyond which the task is not expected to be soluble.

4 Demonstration on Random Sparse Codes A

Mock data were created as follows. The first N lines of $\mathbf{A}$ were set to the identity matrix, and of the remaining bits, a fraction f_A were randomly set to 1. This matrix can be viewed as defining a systematic error correcting code in which the signal $\mathbf{s}$ is transmitted, followed by $(M - N)$ sparse parity checks. Each component of $\mathbf{s}$ was set to 1 with probability 0.5. The vector $\mathbf{t} = \mathbf{A}\mathbf{s} \bmod 2$

was calculated and each of its bits was flipped with noise probability f_n. Four parameters were varied: the vector length N, the number of measurements M, the noise level f_n of the measurements, and the density f_A of the matrix $\mathbf{A}$.

When optimizing θ the following procedure was adopted. The initial condition was $\theta_n = 0, \forall n$. If annealing was used, a sequence of 20 optimizations was performed, with values of β increasing linearly from 0.1 to 1.0. Without annealing, just a single optimization was performed with $\beta = 1$. The optimizers were iterated until the gradient had magnitude smaller than a predefined threshold, or the change in F was smaller than a threshold, or until a maximum number of iterations was completed. At the end, $\mathbf{s}$ was guessed using the sign of each element of θ.

We can calculate a bound on the noise level f_n beyond which the task is definitely not expected to be soluble by equating the capacity of the binary symmetric channel, $(1 - H_2(f_n))$, to the rate of the code, N/M. Here $H_2(f_n)$ is the binary entropy, $H_2(f_n) = -f_n \log_2 f_n - (1 - f_n)\log_2(1 - f_n)$. This bound is indicated on the graphs that follow by a solid horizontal line. For noise levels significantly below this bound we expect the correct solution typically to have significantly greater likelihood than any other $\mathbf{s}$.

Results

For the results reported here, I set N to 50 and M to 500, 1000 and 2000, and ran the synchronous reestimation algorithm multiple times with different seeds, with density f_A varying from 0.05 to 0.50, and error probability f_n varying from 0.05 to 0.45. In each run there are three possible outcomes: 'correct', where the answer is equal to the true vector $\mathbf{s}$; 'wrong', where the answer is not equal to the true vector, and has a smaller likelihood than it; and 'ok', where the algorithm has found an answer with greater likelihood than the true vector (in which case, one cannot complain).

Annealing helps significantly when conjugate gradient optimizers are used, but does not seem to make much difference to the performance of the reestimation algorithms. As was already mentioned, the reestimators work much better than the conjugate gradient minimizers (even with annealing).

Figure 1 shows the outcomes as a function of the weight of $\mathbf{A}$ (x-axis) and error probability (y-axis). There seems to be a sharp transition from solvability to unsolvability. It is not clear whether this boundary constitutes a fundamental bound on what free energy minimization can achieve; performance might possibly be improved by smarter optimizers. Another idea would be to make a hybrid of discrete search methods with a free energy minimizer.

Experiments with larger values of N have also been conducted. The success region looks the same if plotted as a function of of the average number of 1s per row of $\mathbf{A}$ and the noise level.

Table 1. Error rates of error correcting codes using free energy minimization.

a) Sparse random code matrix (section 4). These results use $N = 100$ and $f_A = 0.05$, and various noise levels f_n and $M = 400, 1000, 2000$. The synchronous reestimation optimizer was used without annealing, and with a maximum of 50 gradient evaluations. For every run a new random matrix, signal and noise vector were created. One thousand runs were made for each set of parameters. The capacity of a binary symmetric channel with $f_n = 0.1$, 0.15, 0.2 is 0.53, 0.39, 0.28 respectively.
b) LFSR code (section 5). The number of taps was 5, selected at random. The synchronous reestimation optimizer was used with the annealing procedure described in section 5.

a)

f_n	N	M	Number of errors	Information rate
0.1	100	400	14/1000	0.25
0.1	100	1000	0/1000	0.1
0.15	100	1000	3/1000	0.1
0.2	100	1000	54/1000	0.1
0.2	100	2000	11/1000	0.05

b)

f_n	k	N	Number of errors	Information rate
0.1	100	400	69/1000	0.25
0.15	100	1000	1/1000	0.1
0.2	100	2000	26/1000	0.05

Table 2. BCH codes and Reed-Muller codes.

For each noise level $f_n = 0.1, 0.15, 0.2$, I give n, k, t for selected BCH codes and Reed-Muller codes with $\epsilon^{\text{block}} = P(\text{more than } t \text{ errors}|n) < 0.1$, and rate $> 0.05, 0.03, 0.01$ respectively, ranked by ϵ^{block}. Values of n, k, t from Peterson and Weldon (1972).

$f_n = 0.1$

n	k	t	ϵ^{block}	rate
BCH CODES				
63	10	13	0.003	0.159
127	29	21	0.008	0.228
1023	153	125	0.009	0.150
63	16	11	0.021	0.254
511	85	63	0.037	0.166
REED-MULLER CODES				
128	8	31	8.8e-7	0.063
64	7	15	4.4e-4	0.109
1024	56	127	0.0055	0.055
32	6	7	0.012	0.188
512	46	63	0.038	0.0898
16	5	3	0.068	0.313

$f_n = 0.15$

n	k	t	ϵ^{block}	rate
BCH CODES				
127	8	31	0.002	0.063
511	40	95	0.011	0.078
63	7	15	0.021	0.111
127	15	27	0.022	0.118
255	21	55	0.002	0.082
1023	91	181	0.008	0.089
1023	101	175	0.028	0.099
255	29	47	0.056	0.114
REED-MULLER CODES				
256	9	63	2e-5	0.035
128	8	31	0.0021	0.063
32	6	7	0.096	0.188

$f_n = 0.2$

n	k	t	ϵ^{block}	rate
BCH CODES				
1023	26	239	0.004	0.025
255	9	63	0.028	0.035
511	19	119	0.030	0.037
255	13	59	0.093	0.051
1023	46	219	0.123	0.045
511	28	111	0.152	0.055
REED-MULLER CODES				
1024	11	255	5.7e-5	0.011
512	10	127	0.0034	0.020
128	8	31	0.098	0.063

Potential of this method for error correcting codes

Might an error correcting code using this free energy minimization algorithm be practically useful? I restrict attention to the ideal case of a binary symmetric channel and make comparisons with BCH codes, which are described by Peterson and Weldon (1972) as "the best known constructive codes" for memoryless noisy channels, and with Reed-Muller (RM) codes. These are multiple random error correcting codes that can be characterized by three parameters (n, k, t). The block length is n (this section's M), of which k bits are data bits (this section's N) and the remainder are parity bits. Up to t errors can be corrected in one block. How do the information rate and probability of error of a sparse random code using free energy minimization compare with those of BCH codes?

To estimate the probability of error of the present algorithm I made one thousand runs with $N = 100$ and $f_A = 0.05$ for a small set of values of M and f_n (table 1a). A theoretical analysis will be given, along with a number of other codes using free energy minimization, in (MacKay and Neal 1995).

For comparison, table 2 shows the best BCH and RM codes appropriate for the same noise levels, giving their probability of block error and their rate. I assumed that the probability of error for these codes was simply the probability of more than t errors in n bits. In principle, it may be possible in some cases to make a BCH decoder that corrects more than t errors, but according to Berlekamp (1968), "little is known about...how to go about finding the solutions" and "if there are more than $t + 1$ errors then the situation gets very complicated very quickly."

Comparing tables 1a and 2 it seems that the new code performs as well as or better than equivalent BCH and RM codes, in that no BCH or RM code has both a greater information rate and a smaller probability of block error.

The complexity of the BCH decoding algorithm is $n \log n$ (here, $n \Leftrightarrow M$), whereas that of the FE algorithm is believed to be $w_{A(r)}M$ where $w_{A(r)}$ is the number of 1s per row of $\mathbf{A}$, or at worst $w_{A(r)}MN$. There is therefore no obvious computational disadvantage.

5 Application to a cryptanalysis problem

The inference of the state of a shift register from probabilistic observations of the output sequence is a task arising in certain code breaking problems (Meier and Staffelbach 1989, Anderson 1995).

A cheap 'stream cipher' for binary communication is obtained by sending the bitwise sum modulo 2 of the plain text to be communicated and one bit of a linear feedback shift register (LFSR) of length k bits that is configured to produce an extremely long sequence of pseudo-random bits. This cipher can be broken by an adversary who obtains part of the plain text and the corresponding encrypted message. Given k bits of plain text, the state of the shift register can be deduced, and the entire encrypted message decoded. Even without a piece of plain text, an adversary may be able to break this code if the plain text has high redundancy (for example, if it is an ASCII file containing English words), by guessing part of the plain text.

To defeat this simple attack, the cipher may be modified as follows. Instead of using one bit of the shift register as the key for encryption, the key is defined to be a binary function of a subset of bits of the shift register. Let the number of bits in that subset be h. If this function is an appropriately chosen many-to-one function, it might be hoped that it would be hard to invert, so that even if an adversary obtained a piece of plain text and encrypted text, he would still not be able to deduce the underlying state of the shift register. However, such codes can still be broken (Anderson 1995). Consider a single bit moving along the shift register. This bit participates in the creation of h bits of the key string. It is possible that these h emitted bits together sometimes give away information about the hidden bit. To put it another way, consider the augmented function that maps from $2h-1$ successive bits in the shift register to h successive key bits. Think of the single bit of the preceding discussion as being the middle bit of these $2h-1$ bits; call this middle bit b. Write down all 2^h possible outputs of this mapping, and run through all 2^{2h-1} possible inputs, counting how often each output was produced by an input in which $b=1$, and how often $b=0$. If these two counts are unequal for any of the 2^h outputs, then the occurrence of such an output in the key sequence gives information about the bit b.

In principle, sufficient amounts of such information can be used to break the code. But if computations that scale exponentially with k are assumed not feasible, it may be difficult to use this information. Two algorithms are given by Meier and Staffelbach (1989) for the case where every bit in the shift register sequence has been observed with uniform probability of error; I study the same case here. The methods derived from free energy minimization lead to an algorithm similar to their algorithm B, and thus constitute a solution to the 'open problem' posed at the end of their paper.

Derivation

There are two ways in which the cryptanalysis problem above can be mapped onto this paper's equation $\mathbf{As}+\mathbf{n}=\mathbf{r}$. One might define $\mathbf{s}$ to be the initial state of the shift register, and $\mathbf{r}$ to be the observed noisy sequence, with $\mathbf{A}$ representing the dependence of the mth emitted bit on the initial state, and $\mathbf{n}$ being the noise vector. This representation, however, defines a matrix $\mathbf{A}$ that becomes increasingly dense with 1s as one descends from the top row to the bottom row. It seems likely that a second representation, inspired by the methods of Meier and Staffelbach (1989), will make better use of large quantities of data.

In the second approach, we define $\mathbf{s}$ to be the noise vector. Let the linear feedback shift register's true sequence be $\mathbf{a}_0$, and let the observed noisy sequence be $\mathbf{a}_1=(\mathbf{a}_0+\mathbf{s}) \bmod 2$. Following the notation of Meier and Staffelbach (1989), let the number of bits of state in the shift register be k and let the number of observed bits (*i.e.*, the length of $\mathbf{a}_1$ and $\mathbf{s}$) be N. The algorithms of Meier and Staffelbach (1989) centre on the examination of low-weight parity checks that are satisfied by $\mathbf{a}_0$. If the shift register employs t taps, then for $N \gg k$, a large number $M \simeq N \log(N/2k)$ of relations involving $t+1$ bits of $\mathbf{a}_0$ can be written down. [An even larger number of relations involving $t+2$, $t+3$, etc. bits are also available. An attack based on this method is expected to do best if such relations are also included.] Putting these relations in an $M \times N$ parity matrix $\mathbf{A}$, we have:

$$\mathbf{A}\mathbf{a}_0 \bmod 2 = 0$$

so that

$$\mathbf{A}\mathbf{s} \bmod 2 = \mathbf{r} \tag{8}$$

where $\mathbf{r} = \mathbf{A}\mathbf{a}_1 \bmod 2$ is the 'syndrome' vector listing the values of the parity checks. Equation (8) defines our problem. The vector $\mathbf{s}$ is to be inferred, and, unlike equation (1), there is no noise added to $\mathbf{A}\mathbf{s}$. However, we can define a sequence of problems of the form $\mathbf{A}\mathbf{s} + \mathbf{n} \bmod 2 = \mathbf{r}$ such that the real task is the limiting case in which the noise level goes to zero. Then we can apply the free energy method of this paper to these problems. There are two important differences from the demonstration in section 4. First, there is a non-uniform prior probability over $\mathbf{s}$, favouring low weight $\mathbf{s}$. Second, $\mathbf{A}$ is not a full rank matrix; there are many (2^k) values of $\mathbf{s}$ satisfying equation (8), one for each of the possible initial shift register states. Our task is to find the solution that has maximum prior probability, *i.e.* (in the case of uniform noise level) the $\mathbf{s}$ with the smallest number of 1s.

Demonstration

This demonstration uses random polynomials rather than the special polynomials that are used to make encryption keys, but this is not expected to matter. Test data were created for specified k and N using random taps in the LFSR and random observation noise with fixed uniform probability. The parameter β was initially set to 0.25. For each value of β, a sequential reestimation optimization was run until the decrease in free energy was below a specified tolerance (0.001). β was increased by factors of 1.4 until either the most probable vector under $Q(\mathbf{s}; \theta)$ satisfied all the relations (8), or until a maximum value of $\beta = 4$ was passed. The parameter β_P was set to 1 so that the prior probability was not influenced by the annealing.

In the sequential reestimation procedure the bits $1 \ldots N$ of $\mathbf{s}$ may be updated in any order. I have experimented with various orders: a) the order $1 \ldots N$; b) a random order; c) an order ranked by the 'suspicion' associated with a bit, defined for each bit, following Meier and Staffelbach (1989), to be the fraction of the parity checks in which it is involved that are violated. It seemed plausible that if the most suspect bits are modified first, the algorithm would be less likely to modify good bits erroneously. Experimentally however, I found the difference in performance using these different orderings to be negligible.

Results are shown in figure 2 for $(k, N) = (50, 500)$, $(100, 1000)$ and $(50, 5000)$, using each of the three orderings of bits. A dot represents an experiment. A box represents a success, where the algorithm returned the correct error vector. On each of these graphs I also show three lines. A horizontal line, as in section 4,

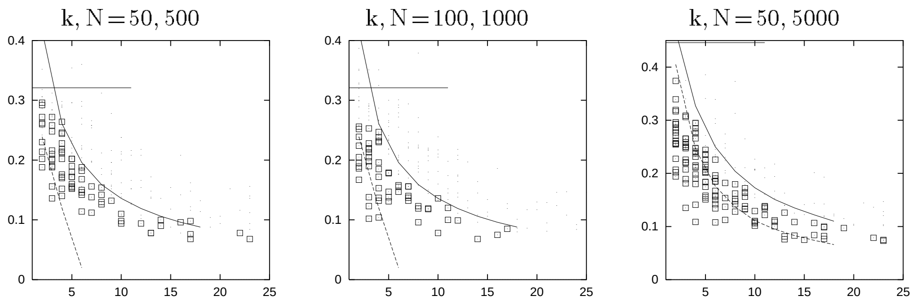

Fig. 2. Results for cryptanalysis problem as a function of number of taps (horizontal axis) and noise level (vertical).

shows the information theoretic bound above which one does not expect to be able to infer the shift register state because the data are too noisy. The two curved lines are from tables 3 and 5 of Meier and Staffelbach (1989), and show (lower line) the limit at which their 'algorithm B appeared to be very successful in most experiments' and (upper line) the theoretical bound beyond which their approach is definitely not feasible. Apparently the algorithm of the present paper not only works fine well beyond the lower line of Meier and Staffelbach, but it frequently finds the correct answer at parameter values right up to the upper line.

Discussion

The forward and backward calculations of the free energy algorithm are similar to calculations in algorithm B of Meier and Staffelbach (1989), but in detail the mapping from $[0,1]^N \rightarrow [0,1]^N$ has a different form. Also, Meier and Staffelbach employ multiple 'rounds' in which the data vector $\mathbf{a}_1$ is changed. This procedure has no analogue in the present algorithm. This algorithm has a well-defined objective function which is guaranteed to decrease during the sequential reestimation algorithm, or which may be minimized by other methods.

A 'bitwise Bayesian' approach to this problem has also been given by Gallager (1963) and Mihaljević and Golić (1993). Their iterative procedure is similar to Meier and Staffelbach's, having two phases in each iteration. In the first phase an inference is made considering each bit individually, assuming a simplified distribution for the other bits. In the second phase a bit-by-bit decision (error-correction) is made. In the present paper, in contrast, the joint posterior distribution of all unknowns given all available information is written down (equation 4), and the iterative procedure optimizes an approximation to this true posterior distribution. No decisions are made until the end of the optimization. These algorithms are similar in that a simplified separable distribution over $\mathbf{s}$ is employed, but the details of this distribution's evolution are different.

The McEliece (1978) public-key cryptosystem depends for its security on the intractability of the general decoding problem $\mathbf{As}+\mathbf{n} = \mathbf{r}$. The present algorithm is not expected to have any application as regards that cryptosystem, as the free energy approximation only appears to be effective when the matrix $\mathbf{A}$ is sparse.

The LFSR system as an error correcting code

The inference of the state of the linear feedback shift register was motivated as a cryptanalytic application, but it can also be viewed as the decoder of a linear error correcting code.

Encoding: A rate k/N and a feedback polynomial with t taps are chosen. A signal of length k bits is used as the initial state of an LFSR. The shift register is iterated for N cycles, the resulting sequence of N bits $\mathbf{a}_0$ being transmitted. This procedure defines a linear code with generator matrix $\mathbf{G}$ in which the first k rows are the identity matrix, and successive rows become increasingly dense.

Decoding: The linear code has an $(N-k)\times N$ parity matrix $\mathbf{H}$ such that for any $\mathbf{a}_0$ generated by the code, $\mathbf{Ha}_0 = 0$. In this special case where the code is generated by an LFSR, $\mathbf{H}$ can be written as a sparse matrix with just weight $t+1$ per row, each row describing the parity check on one cycle of transmission. Furthermore, we can write many more equally sparse parity checks by adding rows of $\mathbf{H}$. The matrix $\mathbf{A}$ of all parity checks of weight $t+1$ is created. This has of order $N\log(N/2k)$ rows. We evaluate the extended syndrome vector $\mathbf{r} = \mathbf{Aa}_1 \bmod 2$. Denoting the noise vector by $\mathbf{s}$, we solve the problem $\mathbf{As} \bmod 2 = \mathbf{r}$ using the free energy minimization method as described earlier. The complexity of this decoding is believed to be $(t+1)N\log(N/2k)$.

As in section 4, I estimated the probability of error of this system by making one thousand runs with $k = 100$ for a small set of values of N and f_n. The results (table 1b) seem comparable with the theoretical performance of BCH codes (table 2), though not as good as the results for sparse random code matrices (table 1a) for $f_n = 0.1$.

6 Conclusion

This paper has derived a promising algorithm for inference problems in modulo 2 arithmetic. Applied to decoding a random sparse error correcting code, this algorithm gives a error/rate trade-off comparable and possibly superior to that of BCH and Reed-Muller codes. For the cryptanalysis problem, the algorithm supersedes Meier and Staffelbach's algorithm B, working close to the theoretical limits given in their paper. The linear feedback shift register system also shows potential for implementation as an error correcting code.

Acknowledgements

I thank Ross Anderson, Radford Neal, Roger Sewell, Robert McEliece and Malcolm MacLeod for helpful discussions, and Mike Cates and Andreas Herz for comments on the manuscript.

Appendix: Pseudo–code

Here follows pseudo-code in C style for the sequential reestimation algorithm. The vector θ of the text is called $\mathbf{x}$ in this appendix. For a more efficient implementation, the matrix $\mathbf{A}$ should be represented in the form of two lists of indices m and n such that $A_{mn} = 1$. I do not include the calculations required for the termination conditions given in the text.

This routine requires as arguments an initial condition $\mathbf{x}$, a matrix $\mathbf{A}$, an observation vector $\mathbf{g}$ as defined in section 1, and a value for β, a prior bias $\mathbf{b}$ as defined in section 3. The final answer is returned in $\mathbf{x}$; a reconstructed binary vector can be obtained by taking the sign of $\mathbf{x}$.

The routine makes use of arrays `q0[n]` and `q1[n]` which contain the probabilities q, and arrays `p0[m][n]` and `p1[m][n]` which contain both the forward and reverse probabilities. Again, for efficient implementation, these arrays should be represented as lists.

This code is free software as defined by the Free Software Foundation.

```
sequential_optimizer          /* arguments :                       */
                              /*           the arrays have indices in */
                              /*           the following ranges:      */
( double *x ,                 /*     x[n] :    x[1] ... x[N]          */
  double **A ,                /*     A[m][n] : A[1][1] ... A[M][N]    */
  double *g ,                 /*     g[m] :    g[1] ... g[M]          */
  double beta ,
  double *bias ,              /*     bias[n] : bias[1] ... bias[N]    */
  int N ,
  int M
)
{
  double *q0 , *q1 ;          /*     q0[n]    : q0[1] ... q0[N]       */
  double **p0 , **p1 ;        /*     p0[m][n] : p0[1][0] ... p0[M][N+1] */

  for ( m = 1 ; m <= M ; m ++ ) {       /* set up boundary conditions */
    p0[m][0]   = 1.0 ; p1[m][0]   = 0.0 ; /*    for forward pass      */
    p0[m][N+1] = 1.0 ; p1[m][N+1] = 0.0 ; /*    and reverse pass      */
  }

  do {
    for ( n = 1 ; n <= N ; n ++ ) {              /* forward pass      */
      q1[n] = 1.0 / ( 1.0 + exp ( - x[n] ) ) ;
      q0[n] = 1.0 / ( 1.0 + exp (   x[n] ) ) ;
      for ( m = 1 ; m <= M ; m ++ ) {
```

```
        if ( A[m][n] == 0  ) {
          p0[m][n] = p0[m][n-1] ;
          p1[m][n] = p1[m][n-1] ;
        }
        else {
          p0[m][n] = q0[n] * p0[m][n-1] + q1[n] * p1[m][n-1] ;
          p1[m][n] = q0[n] * p1[m][n-1] + q1[n] * p0[m][n-1] ;
        }
      }
    }
    for ( n = N ; n >= 1 ; n -- ) {              /* backward pass       */
      gradient_n = 0.0 ;
      for ( m = 1 ; m <= M ; m ++ ) {
        if ( A[m][n] ) {                         /* accumulate gradient */
          gradient_n -= g[m] *
            ( p1[m][n-1] * p1[m][n+1] + p0[m][n-1] * p0[m][n+1] -
              p0[m][n-1] * p1[m][n+1] - p1[m][n-1] * p0[m][n+1]   ) ;
        }
      }
      gradient_n *= beta ;
      gradient_n -= bias[n] ;
      x[n] = - gradient_n ;
      q1[n] = 1.0 / ( 1.0 + exp ( - x[n] ) ) ;
      q0[n] = 1.0 / ( 1.0 + exp (   x[n] ) ) ;
      for ( m = 1 ; m <= M ; m ++ ) {
        if ( A[m][n] == 0 ) {
          p0[m][n] = p0[m][n+1] ;
          p1[m][n] = p1[m][n+1] ;
        }
        else {
          p0[m][n] = q0[n] * p0[m][n+1] + q1[n] * p1[m][n+1] ;
          p1[m][n] = q0[n] * p1[m][n+1] + q1[n] * p0[m][n+1] ;
        }
     }
  } until ( converged ) ;
}
```

References

Aiyer, S. V. B., Niranjan, M. and Fallside, F. (1990). A theoretical investigation into the performance of the Hopfield model, *IEEE Trans. on Neural Networks* **1**(2): 204–215.

Anderson, R. J. (1995). Searching for the optimum correlation attack, *in* B. Preneel (ed.), *Proceedings of 1994 K.U. Leuven Workshop on Cryptographic Algorithms*, Lecture Notes in Computer Science, Springer-Verlag.

Berlekamp, E. R. (1968). *Algebraic Coding Theory*, McGraw-Hill, New York.

Berlekamp, E. R., McEliece, R. J. and van Tilborg, H. C. A. (1978). On the intractability of certain coding problems, *IEEE Transactions on Information Theory* **24**(3): 384–386.

Blake, A. and Zisserman, A. (1987). *Visual Reconstruction*, MIT Press, Cambridge Mass.

Durbin, R. and Willshaw, D. (1987). An analogue approach to the travelling salesman problem using an elastic net method, *Nature* **326**: 689–91.

Feynman, R. P. (1972). *Statistical Mechanics*, W. A. Benjamin, Inc.

Gallager, R. G. (1963). *Low density parity check codes*, number 21 in *Research monograph*, MIT Press, Cambridge, Mass.

Gee, A. H. and Prager, R. W. (1994). Polyhedral combinatorics and neural networks, *Neural Computation* **6**: 161–180.

Hopfield, J. J. and Tank, D. W. (1985). Neural computation of decisions in optimization problems, *Biological Cybernetics* **52**: 1–25.

MacKay, D. J. C. and Neal, R. M. (1995). Error correcting codes using free energy minimization, in preparation.

McEliece, R. J. (1978). A public-key cryptosystem based on algebraic coding theory, *Technical Report DSN 42-44*, JPL, Pasadena.

Meier, W. and Staffelbach, O. (1989). Fast correlation attacks on certain stream ciphers, *J. Cryptology* **1**: 159–176.

Mihaljević, M. J. and Golić, J. D. (1992). A fast iterative algorithm for a shift register initial state reconstruction given the noisy output sequence, *Advances in Cryptology - AUSCRYPT'90*, Vol. 453, Springer-Verlag, pp. 165–175.

Mihaljević, M. J. and Golić, J. D. (1993). Convergence of a Bayesian iterative error-correction procedure on a noisy shift register sequence, *Advances in Cryptology - EUROCRYPT 92*, Vol. 658, Springer-Verlag, pp. 124–137.

Peterson, C. and Soderberg, B. (1989). A new method for mapping optimization problems onto neural networks, *Int. Journal Neural Systems*.

Peterson, W. W. and Weldon, Jr., E. J. (1972). *Error-Correcting Codes*, 2nd edn, MIT Press, Cambridge, Massachusetts.

Van den Bout, D. E. and Miller, III, T. K. (1989). Improving the performance of the Hopfield-Tank neural network through normalization and annealing, *Biological Cybernetics* **62**: 129–139.