A Free Energy Minimization Algorithm for Decoding and Cryptanalysis

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

As submitted to *Electronics Letters*, except including an appendix giving pseudocode.
Submitted November 23, 1994; published 16th March 1995 (vol. 31 no.6)

Abstract

An algorithm is derived for inferring a binary vector $\mathbf{s}$ given noisy observations of $\mathbf{As}$ modulo 2, where $\mathbf{A}$ is a binary matrix. Here, the binary vector is replaced by a vector of probabilities, optimized by free energy minimization. Experiments on the inference of the state of a linear feedback shift register indicate that this algorithm supersedes Meier and Staffelbach's polynomial algorithm.

Index: approximate inference, combinatorial optimization, error correction, stream cipher.

Consider three binary vectors: $\mathbf{s}$ of length N, and $\mathbf{z}$ and $\mathbf{n}$ of length $M \geq N$, related by:

$$(\mathbf{As} + \mathbf{n}) \bmod 2 = \mathbf{z} \tag{1}$$

where $\mathbf{A}$ is a binary matrix. Our task is to infer $\mathbf{s}$ given $\mathbf{z}$ and $\mathbf{A}$, and given assumptions about the statistical properties of $\mathbf{s}$ and $\mathbf{n}$. This problem arises in the decoding of a noisy communication $\mathbf{z}$ which was transmitted using an error-correcting code based on parity checks of the original signal $\mathbf{s}$, and in the inference of the sequence of a linear feedback shift register (LFSR) from a noisy observation of the sequence [1].

I assume that both $\mathbf{s}$ and $\mathbf{n}$ have prior probability distributions that are separable, *i.e.*, $P(\mathbf{s}, \mathbf{n}) = \prod_n P(s_n) \prod_m P(n_m)$. The log probability of the observed data $\mathbf{z}$ as a function of $\mathbf{s}$ (the 'log likelihood') can be written in terms of the noise free vector $\mathbf{t}(\mathbf{s}) = \mathbf{As} \bmod 2$:

$$\log P(\mathbf{z}|\mathbf{s}, \mathbf{A}) \;=\; \sum_m t_m(\mathbf{s})\, g_m + \text{const.} \tag{2}$$

where $g_m \equiv \log[P(n_m = 1)/P(n_m = 0)]$ if $z_m = 0$ and $g_m \equiv -\log[P(n_m = 1)/P(n_m = 0)]$ if $z_m = 1$. The posterior distribution of $\mathbf{s}$ is, by Bayes' theorem:

$$P(\mathbf{s}|\mathbf{z}, \mathbf{A}) = \frac{P(\mathbf{z}|\mathbf{s}, \mathbf{A}) P(\mathbf{s})}{P(\mathbf{z}|\mathbf{A})}. \tag{3}$$

I assume the decoder's aim is to find the most probable $\mathbf{s}$. For large N an exhaustive search over the 2^N possible sequences $\mathbf{s}$ is not feasible. One way to attack such a combinatorial problem is to create a related continuous optimization problem in which the discrete variables are replaced by real variables [2, 3, 4]. Here I derive a continuous representation in terms of a free energy approximation [5] to the awkward posterior distribution (3).

I approximate the true probability distribution by a simpler separable distribution $Q(\mathbf{s};\theta) \equiv \prod_n q_n(s_n;\theta_n)$, parameterized by a vector of parameters θ thus:

$$q_n(s_n=1;\theta_n) = \frac{1}{1+e^{-\theta_n}} \equiv q_n^1; \;\; q_n(s_n=0;\theta_n) = 1 - q_n^1 \equiv q_n^0. \tag{4}$$

The parameters θ are adjusted so as to minimize the variational free energy,

$$F(\theta) = \sum_{\mathbf{s}} Q(\mathbf{s};\theta) \log \frac{Q(\mathbf{s};\theta)}{P(\mathbf{z}|\mathbf{s},\mathbf{A})P(\mathbf{s})}. \tag{5}$$

Even though this summation is over the 2^N discrete values of $\mathbf{s}$, the evaluation of F and its gradient with respect to θ is a tractable problem. It is hoped that an optimized distribution $Q(\mathbf{s};\theta^*)$, where θ^* minimizes F, will give useful information about the original task. Specifically, we might hope that the $\mathbf{s}$ that maximizes $Q(\mathbf{s};\theta^*)$ also maximizes $P(\mathbf{s}|\mathbf{z},\mathbf{A})$.

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 $\mathbf{A}$ (*i.e.*, the number of ones in $\mathbf{A}$). F separates into three terms, $F(\theta) = E_L(\theta) + E_P(\theta) - S(\theta)$, where the 'entropy' is:

$$S(\theta) \equiv -\sum_{\mathbf{s}} Q(\mathbf{s};\theta) \log Q(\mathbf{s};\theta) = -\sum_n \left[q_n^0 \log q_n^0 + q_n^1 \log q_n^1 \right], \tag{6}$$

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

$$E_P(\theta) \equiv -\sum_{\mathbf{s}} Q(\mathbf{s};\theta) \log P(\mathbf{s}) = -\sum_n b_n q_n^1 \tag{7}$$

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:

$$E_L(\theta) \equiv -\sum_{\mathbf{s}} Q(\mathbf{s};\theta) \log P(\mathbf{z}|\mathbf{s},\mathbf{A}) = -\sum_m g_m \sum_{\mathbf{s}} Q(\mathbf{s};\theta)\, t_m(\mathbf{s}) + \text{const}. \tag{8}$$

To evaluate E_L, we thus need the average value of $t_m(\mathbf{s})$ under the separable distribution $Q(\mathbf{s};\theta)$. We can compute this probability for each m by a 'forward' recursion involving a sequence of probabilities $p_{m,\nu}^1$ and $p_{m,\nu}^0$ for $\nu = 1 \ldots N$, 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:

$$\left.\begin{array}{rcl} p_{m,\nu}^1 &=& q_\nu^0 p_{m,\nu-1}^1 + q_\nu^1 p_{m,\nu-1}^0 \\ p_{m,\nu}^0 &=& q_\nu^0 p_{m,\nu-1}^0 + q_\nu^1 p_{m,\nu-1}^1 \end{array}\right\} \text{if } A_{m\nu}=1; \text{ else } \left\{\begin{array}{rcl} p_{m,\nu}^1 &=& p_{m,\nu-1}^1 \\ p_{m,\nu}^0 &=& p_{m,\nu-1}^0 \end{array}\right. \tag{9}$$

with initial condition $p_{m,0}^1 = 0, p_{m,0}^0 = 1$. We obtain: $E_L(\theta) = -\sum_m g_m p_{m,N}^1$.

The derivative of E_L with respect to θ_n can be obtained by evaluating for each m a 'reverse' sequence of probabilities $r_{m,\nu}^1$ and $r_{m,\nu}^0$, defined to be the probabilities that the partial sum $t_m^{\nu N} = \sum_{n=\nu}^{N} A_{mn} s_n \bmod 2$ is equal to 1 and 0 respectively. These can be computed by a recursion for $\nu = N \ldots 1$ like equation (9). Then using the relation

$$p_{m,N}^1 = q_n^0 \left(p_{m,n-1}^1 r_{m,n+1}^0 + p_{m,n-1}^0 r_{m,n+1}^1 \right) + q_n^1 \left(p_{m,n-1}^1 r_{m,n+1}^1 + p_{m,n-1}^0 r_{m,n+1}^0 \right) \tag{10}$$

and defining $d_{mn} = \left(p_{m,n-1}^1 r_{m,n+1}^1 + p_{m,n-1}^0 r_{m,n+1}^0 \right) - \left(p_{m,n-1}^1 r_{m,n+1}^0 + p_{m,n-1}^0 r_{m,n+1}^1 \right)$, we obtain the derivative $\frac{\partial}{\partial\theta_n} E_L(\theta) = -q_n^0 q_n^1 \sum_m g_m d_{mn}$. Thus the derivative of the free energy is:

$$\frac{\partial F}{\partial\theta_n} = q_n^0 q_n^1 \left[\theta_n - b_n - \sum_m g_m d_{mn} \right]. \tag{11}$$

This derivative can be inserted into various gradient–based optimizers. A simple re-estimation optimizer can be written down by noting that everything to the right of θ_n in equation (11) is independent of θ_n, so that the assignment:

$$\theta_n := b_n + \sum_m g_m d_{mn} \tag{12}$$

sets the derivative to zero and is guaranteed to reduce the free energy. This re-estimation equation can be efficiently interleaved with the reverse recursion by evaluating the reverse probabilities $r^1_{m,\nu}$ from $r^1_{m,\nu+1}$ after updating θ_ν.

Optimizers of F can be modified by introducing 'deterministic annealing' techniques [3, 4], in which the non-convexity of the objective function F is switched on gradually by varying an 'inverse temperature' β from 0 to 1. This annealing procedure is intended to prevent the algorithm from running into the local minimum that the initial gradient points towards. We define: $F(\theta, \beta) = \beta E_L(\theta) + E_P(\theta) - S(\theta)$, and perform a sequence of minimizations of this function with successively larger values of β, each starting where the previous one stopped.

The success of the algorithm is expected to depend crucially on the choice of representation of $\mathbf{s}$, with best results if $\mathbf{A}$ is sparse and the true posterior distribution over $\mathbf{s}$ is close to separable.

Computational complexity: The algorithm is expected to take of order 1, or at most N, gradient evaluations to converge, so that the total time taken is of order between $f_A NM$ and $f_A N^2 M$, where f_A is the density of the matrix $\mathbf{A}$. Memory space proportional to $f_A NM$ is required.

Cryptanalysis application

A demonstration of this algorithm on random matrices $\mathbf{A}$ with varying sparsity is given in [6]. Here I describe an application to a cryptanalysis problem, building on the method of Meier and Staffelbach [1]. Assume a linear feedback shift register (LFSR) of length k bits with t feedback taps produces a sequence $\mathbf{a}_0$ of length N bits, and noisy observations $\mathbf{a}_1 = (\mathbf{a}_0 + \mathbf{s}) \bmod 2$ are made (for details see [1],[7]). Here $\mathbf{s}$ is a sparse noise vector of length N. The algorithms in [1] centre on the examination of low-weight parity checks that are satisfied by $\mathbf{a}_0$. For $N \gg k$ we can create a sparse $M \times N$ matrix $\mathbf{A}$ of parity checks such that $\mathbf{A}\mathbf{a}_0 \bmod 2 = 0$, each row of $\mathbf{A}$ having weight $(t+1)$. The noisy sequence $\mathbf{a}_1$ violates some of these parity checks as described by the vector of violations $\mathbf{z} \equiv \mathbf{A}\mathbf{a}_1$. Then the noise vector $\mathbf{s}$ must satisfy:

$$\mathbf{A}\mathbf{s} \bmod 2 = \mathbf{z}. \tag{13}$$

This equation defines our problem. Our knowledge of the noisy observation process defines a prior over $\mathbf{s}$, favouring low weight $\mathbf{s}$. There are many (2^k) values of $\mathbf{s}$ satisfying equation (13), one for each of the possible initial shift register states ($\mathbf{A}$ is not a full rank matrix). Our task is to find among those solutions the $\mathbf{s}$ that has maximum prior probability. Comparing (13) with the generic decoding task of equation (1), one notes that here no noise is added to $\mathbf{A}\mathbf{s}$. However, we can apply the free energy method to a sequence of problems of the form $(\mathbf{A}\mathbf{s} + \mathbf{n}) \bmod 2 = \mathbf{z}$ with increasing inverse temperature β, such that the noise–free task is the limiting case, $\beta = \infty$.

Experimental results

Test data were created for specified k and N using random taps in the LFSR and random observation noise with fixed uniform probability. The matrix $\mathbf{A}$ of all lowest weight parity checks and the vector $\mathbf{z}$ of parity violations were created.

The following procedure was used. The parameter β was increased by factors of 1.4 from a starting value of 0.25. At each value of β, a sequential re-estimation optimization was performed until the decrease in free energy was below a specified tolerance (0.001). At this point, if the most probable vector under $Q(\mathbf{s};\theta)$ satisfied all the relations, the procedure terminated and the reconstructed vector was checked against the true noise vector. Otherwise β was increased, and the next iteration continued from the present value of θ. If a maximum value of $\beta = 4.0$ was passed, the procedure reported a failure.

In the sequential reestimation algorithm the N components of θ may be updated in any order. I have experimented with three orders: left to right, random, and in order of the 'suspicion' associated with a bit, defined for each bit, following [1], to be the fraction of its parity checks that are violated. Experimental results for all orderings were similar. The third ordering was used for the results described here.

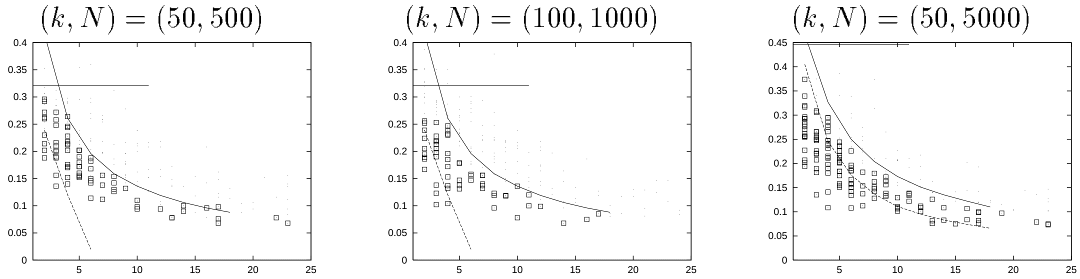

Figure 1: Results for cryptanalysis problem as a function of number of taps (horizontal axis) and noise level (vertical). Squares denote successes.

Results are shown in figure 1 for various (k, N). A dot represents an experiment. A box represents a success, where the algorithm returned the correct error vector. On each graph a horizontal line shows an information theoretic bound above which one does not expect to be able to infer the shift register state because the data is too noisy, and two curved lines, from tables 3 and 5 of ref. [1], show (lower line) the limit up to which Meier and Staffelbach's 'algorithm B appeared to be very successful in most experiments' and (upper line) the theoretical bound beyond which their approach is definitely not feasible. Not only does the algorithm of the present paper apparently work fine well beyond the lower line, but it frequently finds the correct answer at parameter values right up to the upper line.

Conclusion

This paper has derived an algorithm with a well-defined objective function for inference problems in modulo 2 arithmetic. In application to a cryptanalysis problem, this algorithm is similar to Meier and Staffelbach's [1] algorithm B and thus answers their question of whether a derivation could be provided. This algorithm is not identical: the details of the mapping from $[0,1]^N \rightarrow [0,1]^N$ are different, and there is no analogue of Meier and Staffelbach's multiple 'rounds' in which the data vector $\mathbf{a}_1$ is changed. The new algorithm appears to give superior performance and frequently succeeds up to the theoretical limits derived by Meier and Staffelbach.

Acknowledgements

I thank Ross Anderson, Radford Neal and Roger Sewell for helpful discussions, and Michael Cates and Andreas Herz for comments on the manuscript.

Appendix: Pseudo–code

Here follows C–style pseudo–code for the sequential re–estimation algorithm. For a more efficient implementation, the matrix **A** should be represented in the form of two lists, one listing for each m the n such that $A_{mn} = 1$, and the other listing for each n the corresponding m. This code does not show the calculations required for the termination conditions given in the text. The easiest termination condition, and one that is likely to be satisfactory, is to iterate for a fixed number of loops, *e.g.* 10 or 100.

This routine requires as arguments an initial condition $\mathbf{x}$ (NB, the vector θ of the text is called $\mathbf{x}$ in this appendix), a matrix $\mathbf{A}$, an observation vector $\mathbf{g}$ and a prior bias $\mathbf{b}$ as defined in the text and a a value for β (`beta`).

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.

The final answer is returned in the vector $\mathbf{x}$; a reconstructed binary vector can be obtained by taking the sign of $\mathbf{x}$.

```
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
)
{
                            /* internal variables:                     */
  double *q0 , *q1 ;        /*      q0[1] ... q1[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

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

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

[3] A. Blake and A. Zisserman. *Visual Reconstruction.* MIT Press, Cambridge Mass., 1987.

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

[5] R. P. Feynman. *Statistical Mechanics.* W. A. Benjamin, Inc., 1972.

[6] D. J. C. MacKay. A free energy minimization framework for inference problems in modulo 2 arithmetic. In B. Preneel, ed., *Proceedings of 1994 K.U. Leuven Workshop on Cryptographic Algorithms*, Lecture Notes in Computer Science. Springer-Verlag, 1995.

[7] R. J. Anderson. Searching for the optimum correlation attack. In *Proceedings of 1994 K.U. Leuven Workshop on Cryptographic Algorithms*, 1995.