MAXIMUM LIKELIHOOD BINARY SHIFT-REGISTER SYNTHESIS FROM NOISY OBSERVATIONS

Todd K. Moon

Electrical and Computer Engineering Dept., Utah State University
Logan, UT 84322-4120, `Todd.Moon@ece.usu.edu`

ABSTRACT

We consider the problem of estimating the feedback coefficients of a linear feedback shift register (LFSR) based on noisy observations. In the current approach, the coefficients are endowed with a probabilistic model. Gradient ascent updates to coefficient probabilities are computable using recursions developed by means of the EM algorithm. Reduced-complexity approximations are also developed by reducing the number of coefficients propagated at each stage. Applications of this method may include soft decision decoding and blind spread spectrum interception.

1. INTRODUCTION

In this paper we consider the problem of synthesizing the feedback coefficients of a binary linear feedback shift register of fixed length p, given noisy observations of the shift register output and assuming that the distribution of the initial states of the shift register is known. This work is a variation on the seminal work of Massey [1] in two significant ways: the observations are noisy, which means that an exact discrepancy cannot be computed, and the shift register length is fixed, so that the number of coefficients cannot grow to produce a shift register long enough to accommodate the observed sequence. The approach taken in this paper is to regard each feedback coefficient as a Bernoulli $\mathcal{B}(\gamma)$ variable, then estimate the parameters of the Bernoulli variables using a maximum likelihood approach. The EM algorithm [2, 3, 4] is used, leading to expressions for the derivative with respect to the desired parameters which are employed in a steepest-ascent algorithm to increase the EM likelihood function. Such an update is sometimes referred to as a a generalized EM algorithm (GEM).

Despite the recursive updates, the complexity of the algorithm is exponential in the number of coefficients. We therefore consider a reduced complexity algorithm in which only the W most probable paths are extended, in a manner similar to reduced-complexity Viterbi algorithms.

The approach taken here has clear connections to maximum likelihood training of hidden Markov models (HMMs) [5, 6]. The familiar forward probability from HMM training algorithms reappear here, along with some other recursive updates. The work is also related to [7, 8], which considers the problem of inferring the “message” vector $\mathbf{s}$ given observations $\mathbf{r}$ of the form $A\mathbf{s} \oplus \mathbf{n} = \mathbf{r}$, where A is known and $\mathbf{n}$ is Bernoulli noise. This work has been very successfully applied to soft-decision decoding of low-density codes. While we make use of the key insight of the forward and backward probabilities presented in [7], in our noisy LFSR estimation problem, the matrix A is not known in advance (since we do not directly observe the LFSR output). The problem is also related to the sequence inference problems treated in [9, 10, 11].

Potential applications include soft-decision error correction coding (determining the coefficients of an LFSR constitutes a major step in decoding Reed-Solomon codes), intercepting and acquiring code-division multiple access communications, or data compression.

2. PROBLEM STATEMENT

Consider a linear feedback shift register with output y_t given by $y_t = \sum_{i=1}^{p} c_i y_{t-i}$, which is passed through a discrete memoryless channel (DMC) C to produce output d_t, $d_t = C(y_t, n_t)$, where n_t is an independent noise sequence For example, the channel might be a binary symmetric channel (BSC), $d_t = y_t \oplus n_t$ for $n_t \in \{0, 1\}$, or it might be an AWGN channel, $d_t = y_t + n_t, n_t \in \mathbb{R}$. Space limitations preclude providing details for other than the BSC, but extension to the AWGN using the methods here is s-traightforward. The proposed system is diagrammed in figure 1. Given an observed sequence $\{d_t, t = 1, 2, \ldots, T\}$, the problem

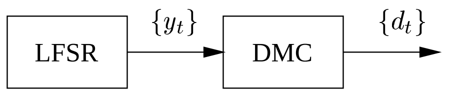


Figure 1: Noisy observations from an LFSR

is to estimate the feedback coefficients $\{c_i, i = 1, 2, \ldots, p\}$. To this end, we model the feedback coefficients as Bernoulli random variables with parameters γ_i defined by $\gamma_i = P[c_i = 1]$. Under the assumption of random coefficients, the LFSR become a Markov model, and, since the outputs are probabilistically determined by means of the BSC, the overall system becomes a hidden Markov model (HMM).

Let $\mathbf{c} = [c_1, c_2, \ldots, c_p]^T$ denote the vector of feedback coefficients, and let $\boldsymbol{\gamma} = [\gamma_1, \gamma_2, \ldots, \gamma_p]^T$ denote the corresponding vector of probabilities. Let $\mathbf{d} = [d_1, d_2, \ldots, d_T]^T$ and $\mathbf{y} = [y_1, y_2, \ldots, y_T]^T$. The state $\mathbf{s}_t$ of the system at time t is determined by the previous p outputs, $\mathbf{s}_t = (y_{t-1}, y_{t-2}, \ldots, y_{t-p}) = (s_{t,1}, s_{t,2}, \ldots, s_{t,p})$. We denote the sequence of states by $\mathbf{s} = [\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_T]$. We observe that the LFSR outputs $\mathbf{y}$ and the state sequence $\mathbf{s}$ are informationally equivalent.

Regardless of the coefficient vector $\mathbf{c}$, the sequence of permissible states of the LFSR must correspond to a DeBruijn sequence [12, p. 16], [13], with the added stipulation that there is only one successor to the all-zero state in the LFSR (it cannot leave the all-zero state). We will denote the set of all allowable state sequences of length T by $\mathcal{D}^T$.

We model each coefficient γ_i according to the logistic

$$\gamma_i = \frac{1}{1+e^{-\theta_i}}, \tag{1}$$

which will lead to an unconstrained optimization problem.

We can formulate the coefficient estimation question as a maximum likelihood (ML) problem: Determine the probabilities $\boldsymbol{\gamma}$ to maximize $P(\mathbf{d}|\boldsymbol{\gamma})$. As a means to the ML solution we will employ the EM algorithm. We identify $\mathbf{x} = (\mathbf{d}, \mathbf{y}) = (\mathbf{d}, \mathbf{s})$ as the complete data set, and $\mathbf{d}$ as the observed data set. Let $\boldsymbol{\gamma}^{[k]}$ denote the estimate of $\boldsymbol{\gamma}$ at the kth iteration of the algorithm, with corresponding parameter $\boldsymbol{\theta}^{[k]}$ determined via (1). For the E-step we form the function

$$\begin{aligned} Q(\boldsymbol{\gamma}|\boldsymbol{\gamma}^{[k]}) &= E[\log P(\mathbf{d},\mathbf{s}|\boldsymbol{\gamma})|\mathbf{d},\boldsymbol{\gamma}^{[k]}] \\ &= \sum_{\mathbf{s}\in\mathcal{D}^T} P(\mathbf{s}|\mathbf{d},\boldsymbol{\gamma}^{[k]})\log P(\mathbf{d},\mathbf{s}|\boldsymbol{\gamma}). \end{aligned} \tag{2}$$

In (2), $\boldsymbol{\gamma}$ is regarded as the unknown vector, to be determined in the M-step by maximizing $Q(\boldsymbol{\gamma}|\boldsymbol{\gamma}^{[k]})$ with respect to $\boldsymbol{\gamma}$.

Noting that $P(\mathbf{d},\mathbf{s}|\boldsymbol{\gamma}) = P(\mathbf{d}|\mathbf{s},\boldsymbol{\gamma})P(\mathbf{s}|\boldsymbol{\gamma})$ and, by the state structure of the LFSR, $P(\mathbf{s}|\boldsymbol{\gamma}) = P(\mathbf{s}_1|\boldsymbol{\gamma})P(\mathbf{s}_2|\mathbf{s}_1,\boldsymbol{\gamma})\cdots P(\mathbf{s}_T|\mathbf{s}_{T-1},\boldsymbol{\gamma})$, we obtain

$$\log P(\mathbf{d},\mathbf{s}|\boldsymbol{\gamma}) = \sum_{t=1}^{T} P(d_t|\mathbf{s}_t,\boldsymbol{\gamma}) + \log P(\mathbf{s}_t|\mathbf{s}_{t-1},\boldsymbol{\gamma}), \tag{3}$$

where we take $P(\mathbf{s}_1|\mathbf{s}_{-1},\boldsymbol{\gamma}) = 0$. We also observe that

$$P(\mathbf{s}|\mathbf{d},\boldsymbol{\gamma}^{[k]}) = \frac{P(\mathbf{d},\mathbf{s}|\boldsymbol{\gamma}^{[k]})}{P(\mathbf{d}|\boldsymbol{\gamma}^{[k]})}, \tag{4}$$

where the denominator is independent of $\boldsymbol{\gamma}$ and may in the future be neglected. Combining (3) and (4) into (2) (and ignoring the denominator) we obtain

$$\begin{aligned} Q(\boldsymbol{\gamma}|\boldsymbol{\gamma}^{[k]}) = &\sum_{\mathbf{s}\in\mathcal{D}^T} P(\mathbf{d},\mathbf{s}|\boldsymbol{\gamma}^{[k]}) \times \\ &\left[\sum_{t=1}^{T}\log P(d_t|\mathbf{s}_t,\boldsymbol{\gamma}) + \log P(\mathbf{s}_t|\mathbf{s}_{t-1},\boldsymbol{\gamma})\right] \end{aligned}$$

The M-step of the EM algorithm leads to $\boldsymbol{\gamma}^{[k+1]} = \arg\max_{\boldsymbol{\gamma}} Q(\boldsymbol{\gamma}|\boldsymbol{\gamma}^{[k]})$. Maximization is approached through a gradient-ascent method, computing an update to $\theta_m^{[k]}$ by $\theta_m^{[k+1]} = \theta_m^{[k]} + \mu\frac{\partial}{\partial\theta_m} Q(\boldsymbol{\gamma}|\boldsymbol{\gamma}^{[k]})$, where μ is the step-size parameter. In what follows, we will develop expressions for $P(d_t|s_t,\gamma)$ and $P(\mathbf{s}_t|\mathbf{s}_{t-1},\gamma)$ in which the parameter γ_m is explicitly apparent, allowing the necessary derivative to be computed.

3. OUTPUT AND STATE PROBABILITIES

Output probabilities The output probabilities $P(d_t|s_t,\gamma)$ are straightforward to express for familiar channel models. For example, for the BSC with crossover probability D, $P(d_t|\mathbf{s}_t,\gamma) = (1-D)P(y_t = d_t|\mathbf{s}_t,\gamma) + DP(y_t = 1-d_t|\mathbf{s}_t,\gamma)$. In general, computation of the output probability thus requires finding the LFSR output probabilities $P(y_t|\mathbf{s},\gamma)$. These probabilities are computed using the forward and backward probabilities introduced in [7]. Let $y_t^{1,m} = \sum_{i=1}^{m} s_{t,i}c_i$ be the *partial forward sum*, and let

$$p_{t,m}^1 = P(y_t^{1,m} = 1|\mathbf{s}_t,\gamma) \qquad p_{t,m}^0 = P(y_t^{1,m} = 0|\mathbf{s}_t,\gamma)$$

be the *forward probabilities*. These probabilities satisfy the recursion

$$\left.\begin{aligned} p_{t,m}^1 &= (1-\gamma_m)p_{t,m-1}^1 + \gamma_m p_{t,m-1}^0 \\ p_{t,m}^0 &= (1-\gamma_m)p_{t,m-1}^0 + \gamma_m p_{t,m-1}^1 \end{aligned}\right\} \text{ if } s_{t,m} = 1$$

$$\left.\begin{aligned} p_{t,m}^1 &= p_{t,m-1}^1 \\ p_{t,m}^0 &= p_{t,m-1}^0 \end{aligned}\right\} \text{ if } s_{t,m} = 0.$$

We similarly define $y_t^{m,p} = \sum_{i=m}^{p} s_{t,i}c_i$ as the *partial backward sum* and let

$$r_{t,m}^1 = P(y_t^{m,p} = 1) \qquad r_{t,m}^0 = P(y_t^{m,p} = 0)$$

be the *reverse probabilities*. These probabilities satisfy a recursion similar to the forward probabilities, with initial condition

$$p_{t,0}^0 = 1 \qquad p_{t,0}^1 = 0 \qquad r_{t,p+1}^0 = 1 \qquad r_{t,p+1}^1 = 0$$

Let

$$\begin{aligned} \Delta_{t,m}^1(\mathbf{s}_t) &= (p_{t,m-1}^0 r_{t,m+1}^0 + p_{t,m-1}^1 r_{t,m+1}^1) \\ \Delta_{t,m}^0(\mathbf{s}_t) &= (p_{t,m-1}^1 r_{t,m+1}^0 + p_{t,m-1}^0 r_{t,m+1}^1) \end{aligned}$$

and

$$\Delta_{t,m}(\mathbf{s}_t) = \Delta_{t,m}^1(\mathbf{s}_t) - \Delta_{t,m}^0(\mathbf{s}_t).$$

Then it can be shown that

$$P(y_t|\mathbf{s}_t,\gamma) = \Delta_{t,m}^{1-y_t}(\mathbf{s}_t) - \gamma_m(-1)^{y_t}\Delta_{t,m}(\mathbf{s}_t)\delta(s_{t,m},1), \tag{5}$$

where $\delta(i,j)$ is the usual Kronecker δ. Then (for the BSC) we obtain

$$\begin{aligned} P(d_t|\mathbf{s}_t,\boldsymbol{\gamma}) = (-1)^{d_t}[&(2D-1)\gamma_m\Delta_{t,m}(\mathbf{s}_t)\delta(s_{t,m},1) \\ &- D\Delta_{t,m}(\mathbf{s}_t)] + \Delta_{t,m}^{1-d_t}(\mathbf{s}_t). \end{aligned} \tag{6}$$

Then (still for the BSC) it follows (with $\tilde{\gamma}_m = \gamma_m(1-\gamma_m)$) that

$$\frac{\partial}{\partial\theta_m}P(d_t|\mathbf{s}_t,\gamma) = (-1)^{d_t}(2D-1)\tilde{\gamma}_m\Delta_{t,m}(\mathbf{s}_t)\delta(s_{t,m},1). \tag{7}$$

State transition probabilities By the state structure of the LFSR, $P(\mathbf{s}|\boldsymbol{\gamma}) = P(\mathbf{s}_1|\boldsymbol{\gamma})P(\mathbf{s}_2|\mathbf{s}_1,\boldsymbol{\gamma})\cdots P(\mathbf{s}_T|\mathbf{s}_{T-1},\boldsymbol{\gamma})$ In the state transition probability $P(\mathbf{s}_t|\mathbf{s}_{t-1},\boldsymbol{\gamma})$, unless $\mathbf{s}_{t-1}$ and $\mathbf{s}_t$ are two adjacent states in a DeBruijn state sequence, this probability is 0. If $\mathbf{s}_{t-1}$ and $\mathbf{s}_t$ are two adjacent states in a DeBruijn sequence, then $P(\mathbf{s}_t|y_{t-1} = y, \mathbf{s}_{t-1},\boldsymbol{\gamma}) = \delta(s_{t,1},y)$. Thus

$$\begin{aligned} P(\mathbf{s}_t|\mathbf{s}_{t-1},\boldsymbol{\gamma}) = P(y_{t-1} = 1|\mathbf{s}_{t-1},\boldsymbol{\gamma})\delta(s_{t,1},1) &+ \\ P(y_{t-1} = 0|\mathbf{s}_{t-1},\boldsymbol{\gamma})\delta(s_{t,1},0)&. \end{aligned}$$

Using (5) we then obtain

$$\frac{\partial}{\partial\theta_m}\log P(\mathbf{s}|\boldsymbol{\gamma}) = \sum_{t=2}^{T}\frac{-(-1)^{s_{t,1}}\tilde{\gamma}_m\Delta_{t-1,m}(\mathbf{s}_{t-1})}{P(\mathbf{s}_t|\mathbf{s}_{t-1},\boldsymbol{\gamma})} \tag{8}$$

From (6), (7) and (8) we can write (after eliminating the nonzero factors $\gamma_m(1-\gamma_m)$ and letting $\Delta_{0,m}(\mathbf{s}_0) = 0$)

$$\begin{aligned} \frac{\partial}{\partial\theta_m}Q(\boldsymbol{\gamma}|\boldsymbol{\gamma}^{[k]}) \propto &\sum_{\mathbf{s}\in\mathcal{D}} P(\mathbf{d},\mathbf{s}|\boldsymbol{\gamma}^{[k]})\sum_{t=1}^{T} C_1(d_t,\mathbf{s}_t,\gamma,m) \\ &+ \sum_{\mathbf{s}\in\mathcal{D}} P(\mathbf{d},\mathbf{s}|\boldsymbol{\gamma}^{[k]})\sum_{t=1}^{T} C_2(\mathbf{s}_{t-1},\mathbf{s}_t,\gamma,m) \end{aligned} \tag{9}$$

where (for the BSC)

$$C_1(d_t, \mathbf{s}_t, \boldsymbol{\gamma}, m) = \frac{(-1)^{d_t}(2D-1)\Delta_{t,m}(\mathbf{s}_t)\delta(s_{t,m},1)}{\nu_1}$$
$$C_2(\mathbf{s}_{t-1}, \mathbf{s}_t, \boldsymbol{\gamma}, m) = \frac{-(-1)^{s_{t,1}}\Delta_{t-1,m}(\mathbf{s}_{t-1})\delta(s_{t-1,m},1)}{\nu_2}.$$

and

$$\nu_1 = (-1)^{d_t}[(2D-1)\gamma_m\Delta_{t,m}(\mathbf{s}_t)\delta(s_{t,m},1) - D\Delta_{t,m}(\mathbf{s}_t)] + \Delta_{t,m}^{1-d_t}(\mathbf{s}_t)$$
$$\nu_2 = \Delta_{t-1,m}^{1}(\mathbf{s}_{t-1}) - \delta(s_{t,1},1)\Delta_{t-1,m}(\mathbf{s}_{t-1}) - (-1)^{s_{t,1}}\gamma_m\Delta_{t-1,m}(\mathbf{s}_{t-1})\delta(s_{t-1,m},1)$$

4. FORWARD AND BACKWARD STATE PROBABILITIES

To compute the sum over all state sequences $\mathbf{s} \in \mathcal{D}^T$ in (9), we introduce the forward and backward state sequence probabilities, similar to those used in training HMMs. Let us define the *forward probability* $\alpha(\mathbf{d}_1^t, \mathbf{j}|\boldsymbol{\gamma}^{[k]})$ as the probability of generating the sequence $d_1, d_2, \ldots, d_t$ and ending up in state[1] $\mathbf{j}$:

$$\alpha(\mathbf{d}_1^t, \mathbf{j}|\boldsymbol{\gamma}^{[k]}) = P(d_1, d_2, \ldots, d_t, \mathbf{s}_t = \mathbf{j}|\boldsymbol{\gamma}^{[k]})$$

Then α has the recursive update

$$\alpha(\mathbf{d}_1^{t+1}, \mathbf{j}|\boldsymbol{\gamma}^{[k]}) = \sum_{\mathbf{i}\in\sigma^-(\mathbf{j})} \alpha(\mathbf{d}_1^t, \mathbf{i}|\boldsymbol{\gamma}^{[k]}) \times P(\mathbf{s}_{t+1} = \mathbf{j}|\mathbf{s}_t = \mathbf{i}, \boldsymbol{\gamma}^{[k]})P(d_{t+1}|\mathbf{s}_{t+1} = \mathbf{j}, \boldsymbol{\gamma}^{[k]}), \quad (10)$$

with initial value

$$\alpha(\mathbf{d}_1^1, \mathbf{i}|\boldsymbol{\gamma}^{[k]}) = \begin{cases} P(d_1|\mathbf{s}_1 = \mathbf{i}, \boldsymbol{\gamma}^{[k]})P(\mathbf{s}_1 = \mathbf{i}) & \mathbf{i} = \mathbf{s}_1 \\ 0 & \mathbf{i} \neq \mathbf{s}_1. \end{cases}$$

The notation $\sigma^-(\mathbf{j})$ indicates the set of precursor states to the state $\mathbf{j}$.

Let us also define the *first forward cost* as

$$\epsilon(\mathbf{d}_1^t, \mathbf{j}, \boldsymbol{\gamma}, m|\boldsymbol{\gamma}^{[k]}) = \sum_{\mathbf{s}_1^{t-1}} P(\mathbf{d}_1^t, \mathbf{s}_1^{t-1}, \mathbf{s}_t = \mathbf{j}|\boldsymbol{\gamma}^{[k]}) \times \sum_{\tau=1}^{t} C_1(d_\tau, \mathbf{s}_\tau, \boldsymbol{\gamma}, m)$$

which can be shown to have the recursive update

$$\epsilon(\mathbf{d}_1^{t+1}, \mathbf{j}, \boldsymbol{\gamma}, m|\boldsymbol{\gamma}^{[k]}) = \sum_{\mathbf{i}\in\sigma^-(\mathbf{j})} P(\mathbf{s}_{t+1} = \mathbf{j}|\mathbf{s}_t = \mathbf{i}, \boldsymbol{\gamma}^{[k]}) \times P(d_{t+1}|\mathbf{s}_{t+1} = \mathbf{j}, \boldsymbol{\gamma}^{[k]})\epsilon(\mathbf{d}_1^t, \mathbf{i}, \boldsymbol{\gamma}, m|\boldsymbol{\gamma}^{[k]}) + \alpha(\mathbf{d}_1^{t+1}, \mathbf{j}|\boldsymbol{\gamma}^{[k]})C_1(d_{t+1}, \mathbf{j}, \boldsymbol{\gamma}, m),$$

with initial value $\epsilon(\mathbf{d}_1^1, \mathbf{j}, \boldsymbol{\gamma}, m|\boldsymbol{\gamma}^{[k]}) = 0$ if $\mathbf{j} \neq \mathbf{s}_1$, or $P(d_1|\mathbf{s}_1, \boldsymbol{\gamma}^{[k]})C_1(d_1, \mathbf{s}_1, \boldsymbol{\gamma}, m)$ if $\mathbf{j} = \mathbf{s}_1$.

[1]Notation: when talking about the value of the state, we will use a bold font, to suggest that the state value is actually a binary p-tuple (a vector).

Let us also define the *second forward cost* as

$$\eta(\mathbf{d}_1^t, \mathbf{j}, \boldsymbol{\gamma}, m|\boldsymbol{\gamma}^{[k]}) = \sum_{\mathbf{s}_1^{t-1}} P(\mathbf{d}_1^t, \mathbf{s}_1^{t-1}, \mathbf{s}_t = \mathbf{j}|\boldsymbol{\gamma}^{[k]}) \times \sum_{\tau=1}^{t} C_2(\mathbf{s}_{\tau-1}, \mathbf{s}_\tau, \boldsymbol{\gamma}, m)$$

with recursive update

$$\eta(\mathbf{d}_1^{t+1}, \mathbf{j}, \boldsymbol{\gamma}, m|\boldsymbol{\gamma}^{[k]}) = \sum_{\mathbf{i}\in\sigma^-(\mathbf{j})} P(\mathbf{s}_{t+1} = \mathbf{j}|\mathbf{s}_t = \mathbf{i}, \boldsymbol{\gamma}^{[k]}) \times P(d_{t+1}|\mathbf{s}_{t+1} = \mathbf{j}, \boldsymbol{\gamma}^{[k]})\eta(\mathbf{d}_1^t, \mathbf{i}, \boldsymbol{\gamma}, m|\boldsymbol{\gamma}^{[k]}) + \sum_{\mathbf{i}\in\sigma^-(\mathbf{j})} P(\mathbf{s}_{t+1} = \mathbf{j}|\mathbf{s}_t = \mathbf{i}, \boldsymbol{\gamma}^{[k]}) \times P(d_{t+1}|\mathbf{s}_{t+1} = \mathbf{j}, \boldsymbol{\gamma}^{[k]})\alpha(\mathbf{d}_1^t, \mathbf{i}|\boldsymbol{\gamma}^{[k]})C_2(\mathbf{i}, \mathbf{j}, \boldsymbol{\gamma}, m).$$

The iteration is initialized with $\eta(\mathbf{d}_1^1, \mathbf{j}, \boldsymbol{\gamma}, m|\boldsymbol{\gamma}^{[k]}) = 0 \ \forall \mathbf{j}, m$. Then

$$\frac{\partial}{\partial\theta_m}Q(\boldsymbol{\gamma}|\boldsymbol{\gamma}^{[k]}) \propto \sum_{\mathbf{j}=0}^{M-1} \epsilon(\mathbf{d}_1^T, \mathbf{j}, \boldsymbol{\gamma}, m|\boldsymbol{\gamma}^{[k]}) + \eta(\mathbf{d}_1^T, \mathbf{j}, \boldsymbol{\gamma}, m|\boldsymbol{\gamma}^{[k]}),$$

where $M = 2^p$. Since we desire this derivative to be zero, we need to find some way of solving for $\boldsymbol{\gamma}$. Rather than solve for $\boldsymbol{\gamma}$ jointly (which would be optimal but more intractable), we will update $\gamma_m, m = 1, 2, \ldots, p$ successively, leaving the other elements of $\boldsymbol{\gamma}$ fixed at the corresponding values of $\boldsymbol{\gamma}^{[k]}$.

In actually performing the computations, normalized values of α, ϵ, and η must be computed, since the computations involve products of probabilities.

5. REDUCING THE COMPUTATIONAL COMPLEXITY

Despite the efficient recurrence, the complexity is still exponential in p, since the quantities α, ϵ and η (or their normalized counterparts) must still be computed for all states and for all time.

In the world of Viterbi algorithm computations, a reduced-complexity, reduced-performance algorithm can be obtained by only retaining a fraction of paths to the state at time t, then extending these. Similarly, we can reduce the complexity, at the expense of some performance, by retaining only a subset of all of the branches. Let W (width) denote the number of branches to retain at each stage. Let S_t denote the set of states that are retained at time t, and let $\sigma^+(S_t)$ denote the successors to those states. Then the algorithm is modified as follows: For each time $t = 2, \ldots, T$:

1. For each $\mathbf{j} \in \sigma^+(S_t)$, update α, ϵ and η. (There are $2\sigma(S_t)$ such states.) The quantities thus obtained will only approximate the true quantities.
2. From these updated values, determine the W states which have the largest state sequence probability α. These W states are selected to form S_{t+1}.
3. Compute updates to the derivatives by

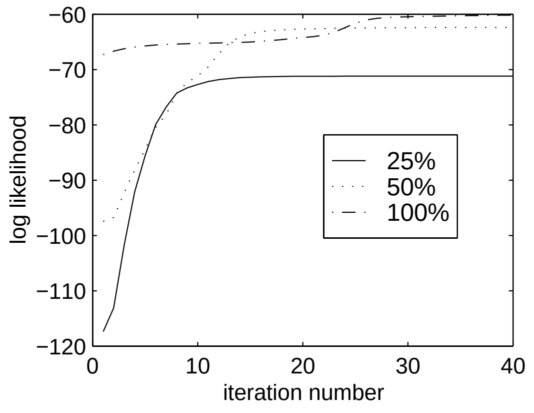
$$\frac{\partial}{\partial\theta_m}Q(\boldsymbol{\gamma}|\boldsymbol{\gamma}^{[k]}) \approx \sum_{\mathbf{j}\in S_{t+1}} \epsilon(\mathbf{d}_1^T, \mathbf{j}, \boldsymbol{\gamma}, m|\boldsymbol{\gamma}^{[k]}) + \eta(\mathbf{d}_1^T, \mathbf{j}, \boldsymbol{\gamma}, m|\boldsymbol{\gamma}^{[k]}),$$

and update the γ vector based on these derivatives.

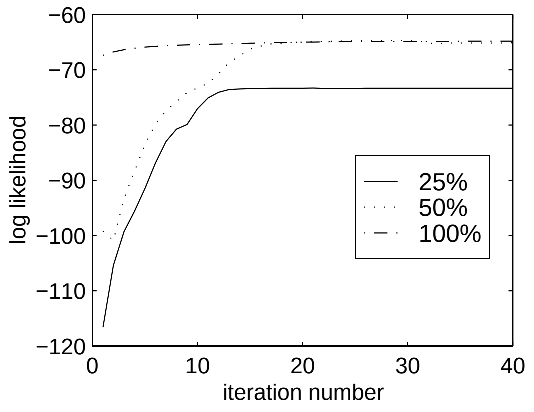
6. ILLUSTRATION OF RESULTS

Space allows for only the briefest of illustrations of performace. Figure 2 illustrates the performance of a test for a BSC with $p = 5$ and $n = 100$, where the computation widths are 25%, 50%, and 100% of the number of states, 32. In part (a), there is no distortion, $D = 0$, while in part (b) $D = 0.1$. The same general behavior is observed, except that there appears to be slightly faster convergence without the distortion. In this particular example, somewhat higher likelihood is obtained with the 100% width computation, but in other experiments, the reduced width computations have provided higher likelihoods.

In both of these cases, and in many other experiments performed, a problem endemic to maximum likelihood techniques is observed: the algorithm frequently converges to a local maximum. Thus this technique might be most useful in a scenario in which the validity of the estimate might be validated, and the algorithm restarted if necessary using different data. One such application might be blind acquisition of spread-spectrum signals, using a RASE technique as described, for example, in [14]. An alternative use might be to use this method as a stage in an iterative soft-decision decoding technique, in which other stages could be used to obtain a useful prior.



(a) No distortion



(b) 10% distortion

Figure 2: Demonstration of the convergence

7. REFERENCES

[1] J. L. Massey, "Shift-Register Synthesis and BCH Decoding," *IEEE Trans. Info. Theory.*, vol. IT-15, no. 1, pp. 122–127, 1969.

[2] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *J. Royal Statiscal Soc., Ser. B*, vol. 39, no. 1, pp. 1–38, 1977.

[3] T. K. Moon and W. C. Stirling, *Mathematical Methods and Algorithms for Signal Processing*. Prentice-Hall, 1999.

[4] T. K. Moon, "The EM Algorithm in Signal Processing," *IEEE Signal Processing Magazine*, vol. 13, pp. 47–60, November 1996.

[5] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proc. IEEE*, vol. 77, pp. 257–286, February 1989.

[6] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *Ann. Math. Stat.*, vol. 41, no. 1, pp. 164–171, 1970.

[7] D. J. C. MacKay, "A free energy minimization framework for inference problems in modulo 2 arithmetic," in *Fast Software Encryption (Proceedings of 1994 K.U. Leuven Workshop on Cryptographic Algorithms)* (B. Preneel, ed.), no. 1008 in Lecture Notes in Computer Science, pp. 179–195, Springer-Verlag, 1995.

[8] D. J. C. MacKay, "Free energy minimization algorithm for decoding and cryptanalysis," *Electronics Letters*, vol. 31, no. 6, pp. 446–447, 1995.

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

[10] M. Mihalvevic and J. Golic, "A fast iterative algorithm for a shift register initial state reconstruction given the noisy output sequence," in *Advances in Cryptology – AUSCRYPT*, vol. 453, pp. 165–175, Springer-Verlag, 1990.

[11] M. Mihalvevic and J. Golic, "Convergence of a bayesian iterative error-correction procedure on a noisy shift register sequence," in *Advances in Cryptology – EUROCRYPT*, vol. 658, pp. 124–137, Springer-Verlag, 1992.

[12] C. Heegard and S. B. Wicker, *Turbo Coding*. Boston: Kluwer Academic, 1999.

[13] S. W. Golomb, *Shift Register Sequences*. San Francisco: Holden–Day, 1967.

[14] R. B. Ward and K. P. Yiu, "Acquisition of pseudonoise signals by recursion-aided sequential estimation," *IEEE Transactions on Communications*, vol. COM-25, no. 8, pp. 784–794, August 1977.