Exploiting symbolic dynamics in chaos coded communications with maximum a posteriori algorithm

F.J. Escribano, L. López and M.A.F. Sanjuán

A known maximum a posteriori (MAP) algorithm is adapted to decode chaotic signals sent over a noisy channel and get a low complexity MAP decoder that could be easily implemented. It is shown that this algorithm is useful for all chaotic encoding frameworks when symbolic dynamics could be applied and that the final bit error rate is better than that obtained with other usually employed maximum likelihood (ML) algorithms of similar complexity.

Introduction: In the past, much attention has been devoted to the task of estimating chaotic sequences affected by noise. Initial efforts were devoted to ML algorithms applied to specific systems such as chaos shift keying [1] or piecewise linear maps (PWLM) [2]. When the chaotic system admits the application of symbolic dynamics [3], a suboptimal ML Viterbi decoding is possible [4, 5].

MAP algorithms have been developed to estimate the initial condition in the case of PWLM [6]. Nevertheless, they are not easily adapted for the purposes of chaotic communications and they are not extendible to other kinds of encodings. As symbolic dynamics allow us to understand the decoder as a first-order Markov process, it is possible to adapt the MAP algorithm known as BCJR (after the initials of the authors) [7]. Although the resulting scheme is normally more complex compared to ML, we show that it is also possible to use it in a lower complexity and practical sliding-window framework that outperforms an ML Viterbi decoding of similar complexity.

Encoding: Though we focus on a very simple chaotic system for brevity’s sake, it will be evident that all the following could be readily applied to other chaotic systems. To encode the binary sequence \( \{b_n\} \), where \( b_n \in \{0, 1\} \) and \( n = 1, \ldots, N \), we will use the known Bernoulli shift map setup [4]. The binary sequence \( \{b_n\} \) is independent and equiprobably distributed. The Bernoulli shift map iterates as follows:

\[
x_{n+1} = f(x_n) = \begin{cases} 2x_n & \text{if } x_n \leq 1/2 \\ 2x_n - 1 & \text{if } x_n > 1/2 \\ \end{cases} \tag{1}
\]

If we define \( x_0 = \sum_{n=0}^{N-1} b_n 2^{-n} \), then the binary sequence is encoded into the chaotic sequence generated by \( x_0 \) and the information can be retrieved following \( b_j = \lfloor x_{j+1} / 2 \rfloor \), where \( \lfloor x \rfloor \) is the floor function, giving the closest integer below \( x \). The probability density function (pdf) of the data generated is \( p(x) = 1/2 \) in \([0, 1] \). In other systems, where it is not possible to get a closed expression for the pdf, it is always possible to use a staircase approximation. The pdf is needed as a priori information for the decoder.

But this kind of encoding is not practical. We can encode blocks of \( D \) bits, following the discretisation \( \tilde{x}^j = \sum_{n=0}^{j-1} b_n 2^{-n} \). This can be seen as controlling the chaotic sequence with small perturbations or as encoding through symbolic dynamics [5]. When \( D \) is about 15–20, the quantisation effects are negligible, the samples keep the desired properties of a chaotic broadband noise-like signal and the discretised pdf is reasonably well approximated by the continuous pdf. We will refer to this chaotic controlled sequence simply as \( \{x_j\} \).

Decoding algorithm: The received signal is \( y_n = x_n + n_a \), where \( n_a \) is an additive white Gaussian noise (AWGN) with zero mean and power \( \sigma^2 \).

The setup for the chaotic BCJR algorithm needs the same definitions as the ML Viterbi algorithm in [4]. We split the \([0, 1]\) interval into a series of non-overlapping intervals \( L \), with limits \( i(P + (i + 1)/P) \) for \( i = 0, \ldots, P - 1 \) and centre in \( c_i = (i+1)/2P \). \( L \) is the number of intervals, is taken as an even number, so that the threshold point \( 1/2 \) is the upper point of one interval and the lower one of another. If we substitute the original sequence by the sequence of intervals where the corresponding sample lies, we get a symbolic representation of the sequence which also conveys the binary information and which can be described as a first-order Markov process, with a corresponding transition matrix \( T \). The term \( t_i \) in this matrix means the transition probability between the interval \( i \) and the interval \( j \), and it depends on the quantisation grid and the form of the encoding. In the case of the Bernoulli shift map, each interval maps exactly into two contiguous intervals with equal probability. Note that this decoding framework can be applied even if symbolic dynamics at the encoder side does not match the symbolic dynamics at the receiver side.

We consider a decoding block of \( L < N \) received symbols \( \{y_{m,n} \} \). We say that the state \( s_k \) at time \( k = 1, \ldots, L \) is \( s_k = 0 \) when \( x_{n+2k} \in L \). To build the BCJR algorithm, we have to calculate the following probability functions \( P, a, \beta \) and \( \lambda \) as stated in [8]:

\[
y_j(l,j) = Pr(s_k = j, y_{n+2k-1} = l) = \sum_i Pr(s_k = j | s_{k-1} = i) Pr(y_{n+2k} = l | \{y_{n,k-1}\})
\]

\[
Pr(y_k = x_{n+2k-1} = i, s_k = j) = \lambda(y_k | x_{n+2k-1} = i, s_k = j)
\]

We substitute the possible values of \( x_k \) (which are \( 2^D \)) by the values of the centre of the intervals \( c_j \) (which are \( P \)). When \( 2^D = P \), we have an instance of Ungerboeck’s trellis coded modulation, and it is not necessary to obtain the pdf to characterise the transitions. To calculate (2) we see that in a transition from \( s_{k-1} = i \) to \( s_k = j \), the only possible quantised encoder output is \( c_j \) and \( Pr(y_k = x_{n+2k-1} = i, s_k = j) = 1 \) when \( x_k \in \{c_j\} \) and is 0 in the rest of cases. \( Pr(s_k = j | s_{k-1} = i) \) is the transition probability \( \lambda(c_j | c_i) \) and \( Pr(y_k = x_{n+2k-1} = i, s_k = j) = 1/(16 \sqrt{2}) \) is the channel output probability. Note that for systems with rate \( R = 1/P \) less than one, we have only to include in this channel metric the \( p \) possible symbols for each transition. The algorithm operates as follows over each block of \( L \) symbols.

Calculate the probability function:

\[
y_j(l,j) = \frac{1}{16 \sqrt{2} \pi} e^{-(y_{n+2k-1} - c_j)^2 / 2 \pi^2} \quad k = 1, \ldots, L \quad i, j = 0, \ldots, P - 1
\]

Forward calculate the probability function:

\[
x_j^f(l,j) = Pr(s_k = j | \{y_{n,2k}\}) = \sum_i x_j^f(i,l) = 1, \ldots, L \quad i, j = 0, \ldots, P - 1
\]

Backward calculate the probability function:

\[
x_j^b(l,j) = Pr(s_k = j | \{y_{n+2k-1} \}) = \sum_i x_j^b(i,l) = 1, \ldots, L \quad i, j = 0, \ldots, P - 1
\]

Finally compute the a posteriori probabilities:

\[
y_j^a(i,j) = Pr(s_k = i, \{y_{n,2k}\}) = x_j^a(l,j) = 1, \ldots, L \quad i, j = 1, \ldots, P
\]

To decode the bit at time \( n \), we take the state \( l_{n+1} \) that maximises the probability \( y_j(i,j) \), and decode according to \( b_k = \lfloor c_{l_{n+1}} + (1/2) \rfloor \). Both \( a_j^f(l,j) \) and \( b_j^f(l,j) \) are usually initialised with \( 1/P \) if \( j = 0, \ldots, P - 1 \). When using this scheme in a sliding-window fashion, in superposing blocks of \( L \) symbols, we take as initial values for \( a_j^f(l,j) \) the result of the forward algorithm \( y_j^f(l,j) \), reserving \( 1/P \) for all \( a_j^f(l,j) \). This ensures that the evidence of the received sequence is propagated to all the decoding blocks. The sliding-window proceeds forward just by taking the following block \( \{y_{n+2k}, \ldots, y_{n+2k+2} \} \). In this way it is not necessary to store and process \( N \times P \) values for the entire sequence. The overhead in calculations is compensated by the saving in memory and by the flexibility of the scheme (i.e. we can decode continuously).

Simulation results: In Fig. 1 we can see the results in terms of BER. \( E_b/N_0 = \sigma^2/2\pi^2 \) and \( c^2 = 1/12 \) is the power of the signal (ideally for \( D \to \infty \)). These results have been obtained with the sliding-window ML Viterbi algorithm [4] and the sliding-window MAP BCJR algorithm. The encoding discretisations are \( D = 5 \) and \( D = 20 \) bits per symbol. Note that the mismatch between the encoding and decoding discretisation \( (D = 20 \) and \( P = 16, 32) \), does not seem to affect the BER, compared with the matched case \( (D = 5 \) and \( P = 32) \).
The sliding-window size is taken as $L = 5$ and $L = 10$. The results for the ML Viterbi and BCJR MAP decoding for the entire block (no sliding-window) are shown for comparison. In each case the MAP sliding-window algorithm yields better results than the ML sliding-window algorithm, while not differing much in complexity (which grows in window algorithm yields better results than the ML sliding-window algorithm) are shown for comparison. In each case the MAP sliding-

---

**Fig. 1** BER performance for different sets of parameters

It has to be noted that with the BCJR algorithm we get probabilistic soft estimates of the data, and this is most useful in some applications, such as the ones applying the iterative decoding philosophy of the so-called turbo codes. We expect that concatenating chaotic encoding systems we could make this kind of system comparable to the practical ones used in communications.

Acknowledgment: We acknowledge financial support from the Spanish Ministry of Science and Technology under project number BFM2003-0381.

© The Institution of Engineering and Technology 2006

16 March 2006

Electronics Letters online no: 20060649
doi: 10.1049/el:20060649

F.J. Escribano and M.A.F. Sanjuán (Nonlinear Dynamics and Chaos Group, Departamento de Matemáticas y Física Aplicadas y Ciencias de la Naturaleza, Universidad Rey Juan Carlos, C/Tulipán s/n, 28933 Móstoles, Spain)

L. López (Laboratorio de Algoritmo Distribuida y Redes, Departamento de Informática, Estadística y Telemática, Universidad Rey Juan Carlos, C/Tulipán s/n, 28933 Móstoles, Spain)

E-mail: francisco.escribano@urjc.es

References


---

Conclusions: We have adapted the MAP BCJR decoding algorithm for a whole class of chaos-channel encoded signals under assumption of symbolic dynamics at the decoder side. We have shown that the BER performance attained by our MAP framework is generally better than the performance achieved with other popular and previously tested algorithms. The principles shown are readily extended to any other kind of encoding with symbolic dynamics.