UNIVERSIDAD DE CANTABRIA

DEPARTAMENTO DE INGENIERÍA DE COMUNICACIONES

TESIS DOCTORAL

ESTIMACIÓN ÓPTIMA DE SECUENCIAS CAÓTICAS CON APLICACIÓN EN COMUNICACIONES

Autor : David Luengo García
Directores : Carlos Pantaleón Prieto
Ignacio Santamaría Caballero

Grupo de Tratamiento Avanzado de Señal

Septiembre de 2006

Capítulo 5

Estimación Computacionalmente Eficiente de Secuencias Caóticas

5.1. Introducción

Los estimadores óptimos mostrados en los dos capítulos anteriores presentan un buen rendimiento, alcanzando el CRLB asintóticamente cuando la varianza del ruido tiende a cero, pero adolecen de un defecto fundamental: su coste computacional crece exponencialmente con el número de observaciones. Esto imposibilita su uso en aplicaciones prácticas, de modo que es necesario desarrollar algoritmos computacionalmente eficientes, pero que al mismo tiempo mantengan las buenas prestaciones de los estimadores ML y Bayesianos. En este capítulo se proponen cuatro estimadores computacional y estadísticamente eficientes de secuencias caóticas. Con este fin, todos los algoritmos intentan encontrar una buena estima de la secuencia simbólica de un modo eficiente, ya que entonces resulta sencillo obtener el estimador ML o MAP de la secuencia caótica.

La estructura del capítulo es la siguiente. En primer lugar, en la Sección 5.2 se analiza el coste computacional de los estimadores desarrollados hasta el momento. A continuación, en la Sección 5.3 se presenta una generalización para cualquier mapa PWL de un algoritmo recursivo, que se denomina FB-ML, propuesto originalmente por Papadopoulos y Wornell para el S-TM. Después, en la Sección 5.4 se estudian los estimadores HC-ML/MAP(k), que consisten en aplicar un umbral duro a las observaciones ruidosas para estimar la secuencia simbólica, modificar los k símbolos con mayor probabilidad de ser erróneos, y finalmente emplear el estimador ML con el itinerario obtenido. En la Sección 5.5 se muestra cómo se puede utilizar una variante del conocido algoritmo E-M, la familia de algoritmos SAGE, para estimar la señal caótica. En la Sección 5.6 se discute el uso del algoritmo de Viterbi (VA) para estimar el itinerario de la secuencia caótica, tras lo cual se puede aplicar el estimador ML estándar para obtener la estima de la señal. Y por último, el capítulo concluye con los resultados obtenidos para los diferentes estimadores eficientes propuestos y su comparación en la Sección 5.7, así como la discusión de los resultados en la Sección 5.8.

5.2. Coste Computacional de los Estimadores Óptimos

En un caso general, en el que se desconoce la secuencia simbólica asociada a una señal caótica, los estimadores óptimos presentados en los dos capítulos anteriores se basan en explorar todos los posibles itinerarios, obtener un estimador para cada uno, y finalmente seleccionar el mejor (ML y MAP) o promediar las estimas (MS). En esta sección se analiza el coste computacional del estimador ML de señales generadas por un mapa PWL, confirmándose su crecimiento exponencial con la longitud de la secuencia para cualquier mapa caótico. El coste de los estimadores Bayesianos es del mismo orden que el del estimador ML, ya que siguen una metodología similar.

5.2.1. Coste Computacional del Estimador ML cuando la Secuencia Simbólica es Conocida

Para realizar un análisis del coste computacional se va a considerar por simplicidad que el punto de referencia se fija en $n = 0$. Esta elección no es relevante, puesto que el coste de los algoritmos para cualquier otro punto de referencia es similar. Asumiendo que se dispone de una partición adecuada del espacio de fases del mapa caótico en regiones precalculada, el Algoritmo 3.3 de estimación ML de $x[0]$ cuando $\mathbf{s}$ es conocido se puede dividir en tres partes:

1. Obtener los coeficientes $A_{\mathbf{s}}^{m,k}$ o equivalentemente la matriz $\mathbf{A}$. Utilizando el Algoritmo 3.1, esta matriz se puede calcular en N iteraciones en las que se determinan sucesivamente sus N columnas. En cada iteración es necesario fijar un valor a uno (el de la diagonal principal) y realizar k multiplicaciones ($k = 2, \ldots, N$) para hallar el resto de elementos de la columna. Por lo tanto, en total esta etapa requiere $(N + 1)(N + 2)/2$ operaciones.

2. Calcular el numerador y denominador de la ecuación (3.33) que proporciona el estimador ML sin limitar. Para ello es necesario obtener $\gamma_{\mathbf{s}}^0[k]$, dado que $\alpha_{\mathbf{s}}^0[k] = A_{\mathbf{s}}^{0,k}$ ya se obtiene directamente a partir de la matriz $\mathbf{A}$. El cálculo de $\gamma_{\mathbf{s}}^0[k]$ se realiza conforme a (3.32), y requiere k productos, $k - 1$ sumas y 1 resta final. Contabilizando la operación extra para cada muestra debida al producto por $\alpha_{\mathbf{s}}^0[k]$, y realizando la suma desde $k = 1$ hasta $k = N$, la obtención del numerador de (3.33) requiere $N(N + 3)$ operaciones. En cuanto al denominador, se trata únicamente de la suma de $N+1$ términos (ya obtenidos previamente) elevados al cuadrado, y por consiguiente el cálculo de $\hat{x}[n]$ conlleva un total de $N(N+5)+1$ operaciones, contando la división final simplemente como una operación más.

3. Finalmente, se debe aplicar un umbral para encontrar $\hat{x}_{\mathrm{ML}}[0]$ e iterar hacia delante para obtener el resto de la secuencia. La primera parte requiere como mucho dos comparaciones y la asignación de un valor a $\hat{x}_{\mathrm{ML}}[0]$. En cuanto a la generación

del resto de la secuencia, en el caso de un mapa PWL supone únicamente una multiplicación y una suma por iteración. En consecuencia, el número total de operaciones de esta última parte será como mucho $2N + 3$.

Juntando el coste computacional de las tres partes del algoritmo, el número total de operaciones requeridas para el cálculo del estimador ML cuando la secuencia simbólica es conocida resulta

$$C_1(N) = \frac{(N+1)(N+2)}{2} + (N(N+5)+1) + (2N+3) = \frac{(N+1)(3N+14)}{2} - 2.$$

Puesto que realmente lo único que nos interesa es el orden de magnitud de las operaciones necesarias, y no el número de operaciones en concreto, siguiendo la notación común a la hora de evaluar el coste computacional de un determinado algoritmo, se va a considerar la *tasa de crecimiento* del número de operaciones [Wilf1986]. En este caso resulta evidente que $C_1(N) = \Theta(N^2)$, y en consecuencia se puede afirmar que el coste computacional del estimador ML cuando el itinerario es conocido crece de manera cuadrática con la longitud de la secuencia de observaciones.

5.2.2. Coste Computacional del Estimador ML cuando la Secuencia Simbólica es Desconocida

En el caso en que se desconoce la secuencia simbólica, el estimador ML exacto consiste básicamente en repetir el procedimiento anterior para cada uno de los $P(N)$ itinerarios posibles. Por lo tanto, el coste computacional total sería

$$C_2(N) = \Theta(P(N)N^2).$$

En el peor caso (todos los itinerarios son válidos) el coste computacional es $C_2(N) = \Theta(M^N N^2)$. Sin embargo, tal y como se ha discutido en la Sección 3.4.3 y mostrado en las tablas 3.6 y 3.8, el número de regiones para una gran cantidad de mapas es mucho menor que M^N. Desafortunadamente, el ritmo de crecimiento de las mismas continúa siendo exponencial, aunque ligeramente suavizado. De hecho, este es uno de los rasgos distintivos de un mapa caótico, cuyo número de secuencias simbólicas válidas de longitud N se puede expresar en general como

$$P(N) = M^{k(N)N}, \tag{5.1}$$

donde $k(N)$ es una función del número de observaciones, comprendida entre 0 y 1, y que depende del mapa caótico y sus parámetros. Esta función indica el ritmo de crecimiento del número de regiones, y por lo tanto el coste computacional del algoritmo.

El valor de $k(N)$ para el TM con los valores de β de la Tabla 3.6 para N entre 6 y 15 se muestra en la Tabla 5.1. El menor valor de $k(N)$ para valores bajos de β supone que se tenga aproximadamente el mismo coste computacional, por ejemplo, para $\beta = 1{,}5$ y $N = 14$, que para $\beta = 1{,}8$ y $N = 11$, o que para $\beta = 2$ y $N = 10$. Sin embargo, la

	$k(N)$									
N	6	7	8	9	10	11	12	13	14	15
$\beta = 1{,}5$	0,875	0,844	0,815	0,794	0,775	0,761	0,747	0,736	0,725	0,716
$\beta = 1{,}6$	0,887	0,863	0,844	0,829	0,816	0,805	0,795	0,786	0,779	0,772
$\beta = 1{,}7$	0,959	0,941	0,924	0,909	0,897	0,886	0,876	0,868	0,861	0,855
$\beta = 1{,}8$	0,968	0,957	0,946	0,937	0,929	0,922	0,917	0,912	0,907	0,903
$\beta = 1{,}9$	0,992	0,987	0,981	0,976	0,971	0,967	0,964	0,961	0,959	0,957
$\beta = 2{,}0$	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000

Tabla 5.1: Índice de crecimiento del número de regiones del TM, $k(N)$, para distintos valores de β y N.

naturaleza exponencial del número de regiones para todos ellos provoca que el coste computacional sufra una explosión en cualquier caso para valores de N medios/altos.

Además, en la Tabla 5.1 se aprecia claramente que $k(N)$ tiende hacia un determinado valor, característico del mapa caótico considerado así como de sus parámetros, conforme N crece. En realidad, resulta trivial demostrar que $k(N)$ está directamente relacionada con la entropía topológica [Beck1993], h_0, mediante la siguiente ecuación:

$$h_0 = \ln M \lim_{N\to\infty} k(N). \tag{5.2}$$

La *entropía topológica* mide la tasa de crecimiento exponencial del número de regiones de un mapa con N. La entropía topológica de un mapa caótico es siempre mayor que cero, lo que implica que $k(N)$ es estrictamente mayor que cero si $M \geq 2$, y que por lo tanto el número de regiones crece exponencialmente con N para cualquier mapa considerado. Es decir, este comportamiento no es típico únicamente de mapas PWL, sino de cualquier mapa caótico.

Utilizando (5.1) y la relación de $k(N)$ con la entropía topológica dada por (5.2), el coste computacional asintótico del Algoritmo 3.6 resulta

$$C_2(N) = \Theta(M^{k(N)N}N^2) \to \Theta(M^{h_0N/\ln M}N^2) = \Theta(e^{h_0N}N^2),$$

Así pues, queda demostrado el crecimiento exponencial de $C_2(N)$ con N para cualquier mapa caótico, de modo que los algoritmos óptimos presentados en los capítulos anteriores resultan inviables para secuencias de longitud media/alta y es necesario desarrollar algoritmos computacional y estadísticamente eficientes como los que se muestran en este capítulo.

5.3. Aproximación Recursiva del Estimador ML para Mapas PWL: Extensión del Algoritmo de Papadopoulos y Wornell

En [Papado1993] y [Papado1995] Papadopoulos y Wornell han propuesto un algoritmo recursivo eficiente para el cálculo del estimador ML de secuencias obtenidas

iterando el S-TM a partir de una cierta condición inicial. Aunque este método es en general subóptimo para otros mapas caóticos, se puede generalizar fácilmente para cualquier mapa PWL, obteniéndose buenos resultados. Este algoritmo encuentra el estimador ML de una secuencia caótica generada por el S-TM en dos etapas:

1. En una primera pasada hacia delante obtiene una secuencia de muestras filtradas, $\hat{x}[n|n]$ $(n = 1, \ldots, N)$, que dependen de la predicción de $x[n]$ dada la información anterior, $\hat{x}[n|n-1] = f(\hat{x}[n-1|n-1])$, y de la información novedosa disponible, dada por la nueva observación, $y[n]$. La recursión se inicializa con $\hat{x}[0|0] = y[0]$.

2. Halla la secuencia ML iterando hacia atrás a partir de $\hat{x}[N|N]$ usando como itinerario el proporcionado por las muestras filtradas, $\hat{x}[n|n]$. Es decir, las N muestras de la secuencia simbólica son $\hat{s}[n] = s_i \Leftrightarrow \hat{x}[n|n] \in E_i$ $(n = 0, \ldots, N)$, y la estima final de cada punto de la señal caótica es $\hat{x}[n|N] = f^{n-N}_{\hat{s}[n],\ \ldots,\ \hat{s}[N-1]}(\hat{x}[N|N])$.

Para poder extender el algoritmo, que se denomina FB-ML ("Forward-Backward" ML) debido a que requiere la iteración hacia delante y hacia atrás del mapa, a cualquier mapa PWL son necesarios los dos lemas presentados a continuación, que son generalizaciones de [Papado1995]. El Lema 5.1 permite intercambiar la iteración hacia atrás y hacia delante, y el Lema 5.2 permite trocar la iteración de un sumatorio de puntos por un sumatorio de la iteración de cada una de las muestras.

Lema 5.1 *Sea $y[k]$ un número real cualquiera correspondiente a la observación ruidosa de la muestra k-ésima de una secuencia caótica generada iterando un mapa PWL unidimensional, sea $x[n]$ el valor real de la iteración n-ésima del mapa, sea $\mathbf{s}_{k:n-1} = [s[k],\ \ldots,\ s[n-1]]^T$ una secuencia simbólica válida arbitraria del mapa PWL, y sean $A^{m,k}_{\mathbf{s}}$ y $B^{m,k}_{\mathbf{s}}$ dados por* (3.14) *y* (3.20) *respectivamente, se cumple que*

$$\left|y[k] - f^{-(n-k)}_{\mathbf{s}_{k:n-1}}(x[n])\right| = \left|\frac{f^{n-k}_{\mathbf{s}_{k:n-1}}(y[k]) - x[n]}{A^{0,n-k}_{\mathbf{s}_{k:n-1}}}\right| = \left|B^{1,n-k}_{\mathbf{s}_{k:n-1}}\left(f^{n-k}_{\mathbf{s}_{k:n-1}}(y[k]) - x[n]\right)\right|. \quad (5.3)$$

Demostración 5.1 *Utilizando* (3.19)*, el término de la izquierda de* (5.3) *puede escribirse como*

$$\begin{aligned}\left|y[k] - f^{-(n-k)}_{\mathbf{s}_{k:n-1}}(x[n])\right| &= \left|y[k] - B^{1,n-k}_{\mathbf{s}_{k:n-1}}x[n] - \sum_{m=1}^{n-k} B^{m,n-k}_{\mathbf{s}_{k:n-m}} b_{s[n-m]}\right| \\ &= \left|\frac{\left(B^{1,n-k}_{\mathbf{s}_{k:n-1}}\right)^{-1} y[k] - \sum_{m=1}^{n-k}\left(B^{1,n-k}_{\mathbf{s}_{k:n-1}}\right)^{-1} B^{m,n-k}_{\mathbf{s}_{k:n-m}} b_{s[n-m]} - x[n]}{\left(B^{1,n-k}_{\mathbf{s}_{k:n-1}}\right)^{-1}}\right|.\end{aligned} \quad (5.4)$$

Notando que $A^{0,n-k}_{\mathbf{s}_{k:n-1}} = \left(B^{1,n-k}_{\mathbf{s}_{k:n-1}}\right)^{-1}$, *que* $\left(B^{1,n-k}_{\mathbf{s}_{k:n-1}}\right)^{-1} B^{m,n-k}_{\mathbf{s}_{k:n-m}} = A^{n-k-(m-1),n-k}_{\mathbf{s}_{n-m+1:n-1}}$, *y con el cambio de variable* $t = n-k-m$ *en el sumatorio del numerador,* (5.4) *resulta*

$$\begin{aligned}\left|y[k] - f^{-(n-k)}_{\mathbf{s}_{k:n-1}}(x[n])\right| &= \left|\frac{A^{0,n-k}_{\mathbf{s}_{k:n-1}} y[k] - \sum_{t=0}^{n-k-1} A^{0,n-k}_{\mathbf{s}_{k:n-1}} b_{s[k+t]} - x[n]}{A^{0,n-k}_{\mathbf{s}_{k:n-1}}}\right| \\ &= \left|\frac{f^{n-k}_{\mathbf{s}_{k:n-1}}(y[k]) - x[n]}{A^{0,n-k}_{\mathbf{s}_{k:n-1}}}\right|,\end{aligned}$$

donde en el último paso se ha usado (3.13). *Con esto queda probada la primera igualdad de* (5.3), *y la segunda resulta evidente dada la relación entre* $A^{0,n-k}_{\mathbf{s}}$ *y* $B^{1,n-k}_{\mathbf{s}}$. □

Nótese que un corolario inmediato del Lema 5.1 es que para cualquier par de puntos x e y se cumple que

$$\left|x - f_s^{-1}(y)\right| = \left|\frac{f_s(x) - y}{a_s}\right|,$$

siendo s la región a la que pertenece x, con $s = 1, \ldots, M$.

Lema 5.2 *Sea* $f(x)$ *un mapa PWL cualquiera con* M *intervalos, y un espacio de fases* $I = [e_0, e_M]$, *definido por* (2.14). *Para cualquier conjunto de* m *constantes,* c_k, *tales que* $\sum_{k=1}^{m} c_k = 1$, *y cualquier conjunto de puntos pertenecientes al espacio de fases del mapa,* x_k, *se cumple que*

$$f_s\left(\sum_{k=1}^{m} c_k x_k\right) = \sum_{k=1}^{m} c_k f_s(x_k),$$

siendo s *uno de los* M *intervalos del mapa caótico.*

Demostración 5.2 *La demostración de este lema es trivial, y se realiza simplemente evaluando los dos lados de la igualdad teniendo en cuenta que* $f_s(x) = a_s x + b_s$. □

Para plantear el algoritmo, en lugar de considerar el problema tratado hasta ahora, estimar la secuencia caótica dado el conjunto de $N+1$ observaciones disponibles, se va a empezar examinando un problema alternativo: obtener la estima ML de cada muestra de la secuencia, $x[n]$ $(n = 0, \ldots, N)$, dada únicamente la secuencia con la observación actual y las anteriores, $\mathbf{y}_{0:n}$. En este caso, la superficie de error resultante es

$$\begin{aligned}J(x[n], \mathbf{s}_{0:n-1}) &= \sum_{k=0}^{n} \left(y[k] - f^{k-n}_{\mathbf{s}_{k:n-1}}(x[n])\right)^2 \\ &= \sum_{k=0}^{n} \left(B^{1,n-k}_{\mathbf{s}_{k:n-1}}\right)^2 \left(f^{n-k}_{\mathbf{s}_{k:n-1}}(y[k]) - x[n]\right)^2,\end{aligned}$$

donde se ha usado el Lema 5.1 para obtener la ecuación final. Esta expresión es cuadrática en $x[n]$, de modo que derivando e igualando a cero se puede obtener fácilmente una estima de $x[n]$ dada la información disponible hasta la muestra n-ésima:

$$\hat{x}[n|n] = \frac{\sum_{k=0}^{n} \left(B_{\hat{\mathbf{s}}_{k:n-1}}^{1,n-k}\right)^2 f_{\hat{\mathbf{s}}_{k:n-1}}^{n-k}(y[k])}{\sum_{k=0}^{n} \left(B_{\hat{\mathbf{s}}_{k:n-1}}^{1,n-k}\right)^2} = \frac{\sum_{k=0}^{n-1} \left(B_{\hat{\mathbf{s}}_{k:n-1}}^{1,n-k}\right)^2 \hat{x}[n|n-1] + y[n]}{\sum_{k=0}^{n} \left(B_{\hat{\mathbf{s}}_{k:n-1}}^{1,n-k}\right)^2}, \tag{5.5}$$

donde la última expresión de (5.5) se ha obtenido usando el Lema 5.2, agrupando e identificando términos. Además, $\hat{x}[n|n-1]$ es la predicción de $x[n]$ con la información disponible hasta el instante $n-1$,

$$\hat{x}[n|n-1] = f_{\hat{s}[n-1]}(\hat{x}[n-1|n-1]),$$

con $1 \leq n \leq N$, y $\hat{\mathbf{s}}_{0:n-1}$ es la secuencia de n símbolos estimada a partir de $\hat{x}[k|k]$ con $k = 0, \ldots, n-1$. Esto es,

$$\hat{s}[n] = i \Leftrightarrow \hat{x}_{\text{FB-ML}}[n|n] \in E_i, \tag{5.6}$$

siendo

$$\hat{x}_{\text{FB-ML}}[n|n] = \begin{cases} e_0, & \hat{x}[n|n] < e_0; \\ \hat{x}[n|n], & e_0 \leq \hat{x}[n|n] \leq e_M; \\ e_M, & \hat{x}[n|n] > e_M. \end{cases} \tag{5.7}$$

Se puede comprobar fácilmente que (5.5) no es más que una generalización del estimador de Papadopoulos y Wornell, ya que para el S-TM se obtiene una expresión idéntica a la proporcionada en [Papado1993, Papado1995]:

$$\hat{x}[n|n] = \frac{(\beta^2-1)\beta^{2n}y[n] + (\beta^{2n}-1)\hat{x}[n|n-1]}{\beta^{2(n+1)}-1}.$$

Nótese que $\hat{s}[n]$ se halla únicamente a partir de la información proporcionada por la observación actual y las anteriores, $\mathbf{y}_{0:n}$. En consecuencia, si su valor óptimo no depende de $\mathbf{y}_{n+1:N}$, entonces $\hat{s}[n] = \hat{s}_{\text{ML}}[n]$ y $\hat{x}_{\text{FB-ML}}[n|n]$ es la estima ML de $x[n]$ dado $\mathbf{y}_{0:n}$, como ocurre para el TM o el S-TM. Sin embargo, para un mapa PWL genérico las observaciones sucesivas sí que pueden aportar información adicional, como ocurre en el caso del SK-TM o del BSK-TM por ejemplo, de modo que $\hat{s}[n] \neq \hat{s}_{\text{ML}}[n]$. En estas ocasiones, $\hat{s}[n]$ es únicamente una aproximación conveniente de $\hat{s}_{\text{ML}}[n]$ obtenida con un coste computacional reducido. El estimador ML debería probar los $P(N)$ itinerarios posibles y seleccionar aquel que minimice la función de coste para encontrar $\hat{\mathbf{s}}_{\text{ML}}$, puesto que todos los símbolos están relacionados y no se pueden ir encontrando recursivamente uno a uno por separado.

En cualquier caso, lo que se desea obtener es la estima ML de la secuencia caótica completa dadas todas las observaciones o una aproximación adecuada en su defecto.

1. Obtener una estima inicial de la secuencia caótica hacia delante (secuencia filtrada, $\hat{x}[n|n]$) utilizando únicamente la información de las muestras anteriores mediante (5.5).

2. Para cada muestra de la secuencia anterior, aplicar un umbral para hallar la estima FB-ML correspondiente, $\hat{x}_{\text{FB-ML}}[n|n]$, usando (5.7).

3. Utilizar como estima de la secuencia simbólica la dada por (5.6) para $n = 0, \ldots, N-1$. Para algunos mapas, como el S-TM, esta es la estima ML del itinerario, mientras que para otros, como el SK-TM, se trata únicamente de una aproximación conveniente.

4. La estima FB-ML de la secuencia caótica se obtiene iterando hacia atrás a partir de $\hat{x}_{\text{FB-ML}}[N|N]$ (que es la estima FB-ML de $x[N]$) con el itinerario obtenido en el paso anterior:
$$\hat{x}_{\text{FB-ML}}[n] = f^{n-N}_{\hat{\mathbf{s}}_{n:N-1}}(\hat{x}_{\text{FB-ML}}[N|N]).$$

Algoritmo 5.1: Estimación ML aproximada de una secuencia caótica generada mediante un mapa PWL cualquiera usando el algoritmo FB-ML.

Esto es equivalente a encontrar la secuencia $\hat{x}[n|N]$ para $0 \leq n \leq N$ siguiendo la notación del algoritmo. Para ello el algoritmo FB-ML procede simplemente iterando hacia atrás a partir de la última muestra estimada usando el itinerario obtenido en la iteración hacia delante,

$$\hat{x}_{\text{FB-ML}}[n] = \hat{x}_{\text{FB-ML}}[n|N] = f^{n-N}_{\hat{\mathbf{s}}_{n:N-1}}(\hat{x}_{\text{FB-ML}}[N|N]).$$

Para mapas como el S-TM en los que $\hat{s}_{\text{ML}}[n] = \hat{s}[n]$ obviamente $\hat{x}_{\text{FB-ML}}[N] = \hat{x}_{\text{ML}}[N]$, y por lo tanto $\hat{x}_{\text{FB-ML}}[n] = \hat{x}_{\text{ML}}[N]$ gracias a la propiedad de invariancia del estimador ML. Para otra clase de mapas PWL en general $\hat{\mathbf{s}} \neq \hat{\mathbf{s}}_{\text{ML}}$, y el estimador FB-ML no coincide con el ML. No obstante, para un mapa PWL cualquiera este algoritmo presenta tres características muy interesantes:

1. Su coste computacional es $\Theta(N^2)$, lo que supone un enorme ahorro con respecto a los algoritmos óptimos.

2. Para un vector de observaciones cualesquiera, si la secuencia simbólica estimada mediante (5.6), $\hat{\mathbf{s}}$, concuerda con la estima ML, $\hat{\mathbf{s}}_{\text{ML}}$, entonces $\hat{\mathbf{x}}_{\text{FB-ML}} = \hat{\mathbf{x}}_{\text{ML}}$.

3. La estima del itinerario dada por (5.6) coincide con la estima ML asintóticamente cuando la SNR tiende a infinito, de modo que $\hat{\mathbf{x}}_{\text{FB-ML}} \to \hat{\mathbf{x}}_{\text{ML}}$.

La implementación de este estimador, ya utilizado en [Panta2000d] para estimar secuencias generadas usando el BSK-TM, se muestra en el Algoritmo 5.1. Se trata de un algoritmo sencillo y que proporciona buenos resultados, especialmente para secuencias largas, alcanzando el CRLB asintóticamente cuando la SNR tiende a infinito.

5.4. Estima del Itinerario Aplicando un Umbral Duro a las Observaciones: HC-ML/MAP

En esta sección se presentan dos algoritmos heurísticos que permiten establecer un compromiso entre coste computacional y rendimiento: el HC-ML ("Hard Censoring" ML) y el HC-MAP. Ambos algoritmos se basan en obtener una estima inicial de la secuencia simbólica aplicando un umbral duro sobre las observaciones ruidosas, y después posiblemente tratar de modificar dicho itinerario inicial buscando otro mejor. Por último, se emplea el estimador ML/MAP usando el itinerario final seleccionado.

El algoritmo HC-ML se propuso en primer lugar en [Panta2000d] para estimar secuencias generadas iterando el BSK-TM, aunque se puede extender inmediatamente a cualquier mapa PWL. Considérese un mapa PWL cualquiera con M regiones y un espacio de fases $I = [e_0,\ e_M]$. En su versión básica, el HC-ML consiste simplemente en obtener una secuencia de observaciones recortadas usando (4.64),

$$y_T[n] = \begin{cases} e_0, & y[n] < e_0; \\ y[n], & e_0 \leq y[n] \leq e_M; \\ e_M, & y[n] > e_M, \end{cases} \tag{5.8}$$

con $1 \leq k \leq N$, utilizarlas para obtener la estima del itinerario aplicando un umbral duro dentro de los límites de las diferentes regiones,

$$\hat{s}[n] = i \Leftrightarrow y_T[n] \in E_i, \tag{5.9}$$

y finalmente aplicar el estimador ML dado por el Algoritmo 3.3 usando dicho itinerario.

Por un lado, el estimador ML exacto ofrece buenas prestaciones, pero con un coste computacional muy elevado. Por otra parte, el estimador HC-ML básico presenta un coste computacional muy reducido, $\Theta(N^2)$, pero con unas prestaciones mucho menores, especialmente para SNRs intermedias. No obstante, el algoritmo HC-ML básico se puede modificar para alcanzar una solución de compromiso entre ambos. Este algoritmo modificado, denominado HC-ML(k), se basa en utilizar como punto de partida el itinerario obtenido a partir de las observaciones ruidosas, y posteriormente modificar aquellas k muestras de la secuencia simbólica que presenten una mayor probabilidad de ser erróneas. De este modo, se deben calcular como mucho M^k estimadores ML (pueden existir itinerarios inválidos), y el coste computacional en el peor caso es $\Theta(M^k N^2)$. Obviamente, HC-ML(0) es el algoritmo HC-ML básico (es decir, sin modificar ninguna muestra del itinerario), y HC-ML(N) es el estimador ML exacto, obtenido probando las $P(N)$ secuencias simbólicas válidas.

Lo único que falta aún por dilucidar es qué subconjunto de k símbolos de entre los N existentes presentan una mayor probabilidad de ser erróneos, y por lo tanto se deben explorar para tratar de hallar la secuencia simbólica óptima. Para estudiar el problema, considérese un mapa PWL unimodal, cuyo espacio de fases se puede dividir en dos regiones: $E_1 = [e_0,\ e_1)$ y $E_2 = [e_1,\ e_2]$. Las posibilidades de que exista un error

en la muestra n-ésima del itinerario son, bien que $s[n] = 1$ y $\hat{s}[n] = 2$, o bien que $s[n] = 2$ y $\hat{s}[n] = 1$. En el primer caso, la probabilidad de error es

$$\begin{aligned} \Pr(\hat{s}[n] = 2 | s[n] = 1) &= \Pr(y_T[n] \in E_2 | x[n] \in E_1) \\ &= \Pr(e_1 \leq y_T[n] \leq e_2 | e_0 \leq x[n] < e_1) \\ &= \Pr(w[n] \geq e_1 - x[n]) \\ &= \frac{1}{2} \operatorname{erfc}\left(\frac{e_1 - x[n]}{\sqrt{2\sigma^2}}\right), \end{aligned} \tag{5.10}$$

donde la última expresión es debida a que el ruido es Gaussiano, y $\operatorname{erfc}(x)$ es la función de error complementaria [Abramo1965]

$$\operatorname{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty \exp(-t^2)\, dt = 1 - \operatorname{erf}(x). \tag{5.11}$$

De igual modo, para el caso en que $s[n] = 2$ y $\hat{s}[n] = 1$ se encuentra que

$$\Pr(\hat{s}[n] = 1 | s[n] = 2) = \frac{1}{2} \operatorname{erfc}\left(\frac{x[n] - e_1}{\sqrt{2\sigma^2}}\right).$$

Nótese que en ambos casos la probabilidad de que el símbolo n-ésimo de la secuencia estimado inicialmente, $\hat{s}[n]$, sea distinto del real, $s[n]$, depende inversamente de $d[n] = |x[n] - e_1|$, ya que $\operatorname{erfc}(x)$ es una función monótona estrictamente decreciente. Es decir, obviamente la probabilidad de cometer un error en $\hat{s}[n]$ es menor cuanto mayor es la distancia de $x[n]$ al punto que delimita las dos regiones de la partición, e_1. A continuación, utilizando la desigualdad triangular [Abramo1965], se encuentra que

$$|y[n] - e_1| - |w[n]| \leq d[n] \leq |y[n] - e_1| + |w[n]|,$$

que relaciona $d[n]$ con la distancia de la observación, $y[n]$, al umbral, e_1. De modo que cuando el ruido es pequeño,

$$\Pr(\hat{s}[n] \neq s[n]) = \frac{1}{2}\operatorname{erfc}\left(\frac{d[n]}{\sqrt{2\sigma^2}}\right) \simeq \frac{1}{2}\operatorname{erfc}\left(\frac{|y[n] - e_1|}{\sqrt{2\sigma^2}}\right), \tag{5.12}$$

siendo la segunda expresión exacta asintóticamente cuando la varianza del ruido tiende a cero. En consecuencia, para un mapa PWL con dos niveles se deben probar aquellos k símbolos para los cuales la distancia de $y[n]$ a e_1 sea menor, ya que sin otra información a priori son los que presentan una mayor probabilidad de ser erróneos.

Para un mapa PWL con M intervalos la situación es similar, aunque ahora se deben considerar dos clases de intervalos. Para aquellas regiones situadas en los extremos de la partición, las regiones "exteriores" E_1 y E_M, la probabilidad de error viene dada por (5.12), con $d[n] = |x[n] - e_1|$ para E_1 y $d[n] = |x[n] - e_{M-1}|$ para E_M. El resto de los intervalos de la partición son "interiores", y su probabilidad de error resulta

$$\begin{aligned} \Pr(\hat{s}[n] = i | s[n] \neq i) &= \frac{1}{2}\left|\operatorname{erfc}\left(\frac{d_{i-1}[n]}{\sqrt{2\sigma^2}}\right) - \operatorname{erfc}\left(\frac{d_i[n]}{\sqrt{2\sigma^2}}\right)\right| \\ &\simeq \frac{1}{2}\left|\operatorname{erfc}\left(\frac{|y[n] - e_{i-1}|}{\sqrt{2\sigma^2}}\right) - \operatorname{erfc}\left(\frac{|y[n] - e_i|}{\sqrt{2\sigma^2}}\right)\right|, \end{aligned} \tag{5.13}$$

siendo $d_{i-1}[n] = |x[n] - e_{i-1}|$ y $d_i[n] = |x[n] - e_i|$. De nuevo la última aproximación es exacta asintóticamente, e indica que la probabilidad de error es directamente proporcional a la distancia de $y[n]$ al umbral. La diferencia entre (5.12) y (5.13) estriba en que las regiones "interiores" tienen una región adyacente tanto por la izquierda como por la derecha, y por lo tanto aparecen dos umbrales que deben considerarse. No obstante, cuando $y[n]$ se encuentre muy desplazada con respecto al centro de la región, el más cercano predomina a la hora de calcular la probabilidad de error, y (5.13) se puede aproximar por (5.12) usando $d[n] = \text{mín}\{d_{i-1}[n], d_i[n]\}$.

En conclusión, el algoritmo HC-ML(k) funciona modificando aquellos k símbolos cuya probabilidad de error, dada por (5.12) o (5.13), sea mayor. Nótese que en este punto existen dos posibilidades: probar todas las combinaciones de símbolos válidos para los k modificados, o utilizar únicamente aquel símbolo más probable distinto del estimado inicialmente (esto es, el correspondiente a la región más cercana a $y[n]$ distinto de $\hat{s}[n]$). El primer algoritmo, HC-ML1(k), requiere el cálculo de como mucho M^k estimadores ML con un itinerario dado, y coincide con el ML para $k = N$. Para el segundo algoritmo, HC-ML2(k), sólo es necesario obtener un máximo de 2^k estimadores ML, pero puede no coincidir con el estimador ML, incluso para $k = N$. Obviamente, para mapas con dos intervalos el algoritmo HC-ML1(k) y el HC-ML2(k) son idénticos, y en cualquier caso ambos coinciden asintóticamente cuando la SNR tiende a infinito. El esquema básico de este estimador se muestra en el Algoritmo 5.2, y su rendimiento en la Sección 5.7.

Por último, el algoritmo HC-MAP(k) es idéntico al HC-ML, sólo que buscando ahora aquel de los M^k (para el HC-MAP1(k)) o 2^k (para el HC-MAP2(k)) itinerarios que minimice $J_{\text{MAP}}(x[0], \mathbf{s})$ en lugar de $J(x[0], \mathbf{s})$. Evidentemente, para $k = 0$ la estima HC-ML coincide con la HC-MAP, puesto que únicamente se considera el itinerario obtenido directamente a partir de las observaciones ruidosas.

5.5. Implementación Iterativa del Estimador ML: Algoritmos E-M y SAGE

Aprovechando el paralelismo existente entre el problema de la estimación de la secuencia simbólica y el de la detección multiusuario, ya reseñado en la Sección 3.4.4.1, en esta sección se considera el uso de dos técnicas de eficacia probada en este último caso para la estimación del itinerario de la señal caótica: los algoritmos E-M y SAGE.

5.5.1. Introducción a los Algoritmos E-M y SAGE

Aunque ya se habían descubierto y empleado con anterioridad muchas de sus ideas principales, el algoritmo E-M ("Expectation-Maximization") debe su formulación actual a Dempster, Laird y Rubin [Demps1977]. Se trata de un algoritmo iterativo que trata de obtener el estimador ML de un conjunto de parámetros, $\boldsymbol{\theta} = [\theta_1, \ \ldots, \ \theta_p]^T$, a partir de las observaciones, $\mathbf{y} = [y[0], \ \ldots, \ y[N]]^T$. En muchos problemas la maximiza-

1. Obtener una secuencia de observaciones pertenecientes al espacio de fases del mapa, $y_T[n]$, aplicando un umbral como en (5.8).
2. Realizar una estima inicial de la secuencia simbólica, $\hat{s}[n]$, a partir de $y_T[n]$ usando (5.9).
3. Calcular la distancia de cada observación a las regiones adyacentes, y estimar la probabilidad de que $\hat{s}[n]$ sea errónea mediante (5.12) para las regiones "exteriores" (E_1 y E_M) y mediante (5.13) para las regiones "interiores" ($E_2,\ \ldots,\ E_{M-1}$).
4. Seleccionar aquellas k muestras cuya probabilidad de error sea mayor.
5. Modificar la secuencia simbólica inicial, $\hat{s}[n]$, del siguiente modo:
   - 5.1. Para el algoritmo HC-ML1(k): Construir todos los itinerarios de longitud N válidos usando todas las combinaciones posibles de símbolos para cada una de las k muestras seleccionadas (M^k como mucho).
   - 5.2. Para el algoritmo HC-ML2(k): Construir todos los itinerarios de longitud N válidos usando para cada una de las k muestras seleccionadas únicamente el símbolo dado por $\hat{s}[n]$ y aquel correspondiente a la región adyacente más cercana a $y[n]$ (2^k como mucho).
6. Para cada itinerario válido obtenido en el paso anterior, calcular la estima ML de la secuencia caótica mediante el Algoritmo 3.3.
7. Elegir como estima ML de la secuencia aquella que proporcione un menor error cuadrático con las observaciones. Esto es, la que minimice (3.8).

Algoritmo 5.2: Estimación ML aproximada de una secuencia caótica generada mediante la iteración de un mapa PWL cualquiera usando los algoritmos HC-ML1(k) o HC-ML2(k).

ción directa de la función de verosimilitud, $p(\mathbf{y};\boldsymbol{\theta})$, resulta complicada, y no es posible encontrar de manera exacta y cerrada el estimador ML. En estos casos, el algoritmo E-M aborda el problema suponiendo que el valor óptimo de $\boldsymbol{\theta}$ no depende únicamente de las observaciones disponibles, denominadas *conjunto incompleto de datos*, sino también de un *conjunto de datos ocultos* ("missing data"), que no son directamente observables, $\boldsymbol{\phi}$. A partir del conjunto incompleto de datos y el de datos ocultos se puede construir el *conjunto completo de datos*, $\mathbf{z} = \mathbf{f}(\mathbf{y}, \boldsymbol{\phi})$, que permite formular el estimador ML de manera alternativa como

$$\begin{aligned}
\hat{\boldsymbol{\theta}}_{\mathrm{ML}} &= \arg\max_{\boldsymbol{\theta}} \ \ln p(\mathbf{y};\boldsymbol{\theta}) \\
&= \arg\max_{\boldsymbol{\theta}} \ \{\ln p(\mathbf{z},\mathbf{y};\boldsymbol{\theta}) - \ln p(\mathbf{z}|\mathbf{y};\boldsymbol{\theta})\} \\
&= \arg\max_{\boldsymbol{\theta}} \ \left\{\mathrm{E}_{\mathbf{z}|\mathbf{y}}(\ln p(\mathbf{z},\mathbf{y};\boldsymbol{\theta})) - \mathrm{E}_{\mathbf{z}|\mathbf{y}}(\ln p(\mathbf{z}|\mathbf{y};\boldsymbol{\theta}))\right\}, \qquad (5.14)
\end{aligned}$$

donde $\mathrm{E}_{\mathbf{z}|\mathbf{y}}(\cdot)$ denota la esperanza matemática con respecto a $p(\mathbf{z}|\mathbf{y};\boldsymbol{\theta})$, que no afecta a la formulación original del estimador ML y que permite eliminar la dependencia de (5.14) con los datos ocultos, no observables.

Aunque la última ecuación de (5.14) puede resultar más difícil de maximizar que la expresión original, el principal resultado teórico relacionado con el algoritmo E-M garantiza que el incremento de $\mathrm{E}_{\mathbf{z}|\mathbf{y}}(\ln p(\mathbf{z},\mathbf{y};\boldsymbol{\theta}))$ provoca un aumento de la función de log-verosimilitud, $\ln p(\mathbf{y};\boldsymbol{\theta})$ [Demps1977]. En consecuencia, el algoritmo E-M procede a maximizar dicho término llevando a cabo únicamente dos pasos de manera iterativa:

1. Etapa E (Expectativa o Esperanza Matemática): Calcula la esperanza matemática de $\ln p(\mathbf{z},\mathbf{y};\boldsymbol{\theta})$ con respecto a $p(\mathbf{z}|\mathbf{y};\hat{\boldsymbol{\theta}}_{\mathrm{EM}}^{k-1})$, $Q(\boldsymbol{\theta},\hat{\boldsymbol{\theta}}_{\mathrm{EM}}^{k-1})$, siendo $\hat{\boldsymbol{\theta}}_{\mathrm{EM}}^{k-1}$ la estima del vector de parámetros obtenida en la iteración anterior.

2. Etapa M (Maximización): Encuentra la estima k-ésima del vector de parámetros, $\hat{\boldsymbol{\theta}}_{\mathrm{EM}}^{k}$, como aquel valor de $\boldsymbol{\theta}$ que maximiza la esperanza matemática obtenida en la etapa anterior. Esto es, $\hat{\boldsymbol{\theta}}_{\mathrm{EM}} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta},\hat{\boldsymbol{\theta}}_{\mathrm{EM}}^{k-1})$.

La principal ventaja del algoritmo E-M frente a otros métodos iterativos usados para maximizar $p(\mathbf{y};\boldsymbol{\theta})$ es que, bajo condiciones poco restrictivas, su convergencia está asegurada. Se puede demostrar que, si $Q(\boldsymbol{\theta},\hat{\boldsymbol{\theta}}_{\mathrm{EM}}^{k-1})$ es continua tanto en $\boldsymbol{\theta}$ como en $\hat{\boldsymbol{\theta}}_{\mathrm{EM}}^{k-1}$, el algoritmo converge a un punto estacionario de la función de log-verosimilitud [Wu1983]. Puesto que además la etapa M garantiza que en cada iteración se incrementa la verosimilitud, este punto estacionario debe ser necesariamente un máximo [Demps1977]. Desafortunadamente, aunque el algoritmo E-M proporciona buenos resultados en general, también presenta algunos inconvenientes importantes:

1. Cuando la función de verosimilitud es multimodal el algoritmo E-M converge en general a un máximo local cuyo valor depende de la condición inicial.

2. En algunos problemas puede resultar difícil hallar una expresión cerrada de $Q(\boldsymbol{\theta},\hat{\boldsymbol{\theta}}_{\mathrm{EM}}^{k-1})$ en la etapa E y/o llevar a cabo su maximización en la etapa M.

3. Aunque en algunos casos existe un conjunto de datos ocultos "natural", habitualmente la elección de los mismos es arbitraria y condiciona en gran medida el rendimiento del algoritmo. En general, cuanta mayor información adicional aporten los datos ocultos más sencilla resulta la etapa M, aunque a costa de añadir complejidad a la etapa E y ralentizar la convergencia del algoritmo [Hero1993, Fessle1993a].

Se han propuesto numerosos métodos para limitar la complejidad del algoritmo E-M a la vez que se acelera su velocidad de convergencia (véase por ejemplo [Meng1997] para una revisión), pero en esta Tesis sólo se considera uno de ellos: las técnicas SAGE. La

familia de algoritmos SAGE ("Space-Alternating Generalized" EM) fue propuesta por Fessler y Hero para resolver problemas en los que es factible actualizar secuencialmente subconjuntos del vector de parámetros [Fessle1993b, Fessle1994], aplicándose rápidamente a problemas de detección multiusuario en comunicaciones digitales [Nelson1996].

La idea básica consiste en introducir en cada iteración una etapa D (Definición) previa a la etapa E, en la que se determina el conjunto de parámetros actualizados en la misma mediante una iteración del algoritmo E-M estándar. Esta clase de métodos consiguen resolver dos de los principales inconvenientes del algoritmo E-M: su lenta convergencia y las complicadas etapas de maximización. Por un lado, permiten desacoplar la actualización de los distintos parámetros, simplificando la etapa de maximización. Y por otro lado, gracias al uso de un espacio completo de datos menos informativo propician una mejora en la tasa de convergencia asintótica y un mayor incremento de la verosimilitud en las primeras iteraciones [Fessle1994].

Para finalizar, a pesar de que no existe ninguna regla formal respecto a la elección del subconjunto de parámetros en cada iteración, un buen algoritmo debe escoger todos los parámetros del conjunto con suficiente frecuencia para que su rendimiento y velocidad de convergencia sean adecuados. Por ejemplo, si existen un total de p parámetros $(\theta_1,\ \ldots,\ \theta_p)$ y se actualiza sólo uno en cada iteración, la manera más sencilla y efectiva de seleccionarlos consiste en irlos escogiendo de manera consecutiva, eligiendo de nuevo el primero después de haber modificado el último de ellos [Fessle1994]. De este modo se garantiza que tras cada p iteraciones del algoritmo se ha recorrido el conjunto de parámetros completo.

5.5.2. Estimación de Señales Caóticas con el Algoritmo E-M

En esta sección se considera la aplicación del algoritmo E-M para la estimación de señales caóticas. En principio, la naturaleza fractal de las superficies de error asociadas a cualquier método de estimación de señales caóticas no augura un buen funcionamiento de algoritmos iterativos como el E-M o el SAGE, cuya solución se correspondería en general con un mínimo local de la función de coste. Sin embargo, la disponibilidad de una buena condición inicial que coincide asintóticamente con la secuencia caótica real (la estima HC-ML de $\mathbf{s}$ y $x[0]$), va a compensar en gran medida este hecho, permitiendo obtener estimas mediante el algoritmo E-M que coinciden con el estimador ML cuando la SNR tiende a infinito. En este sentido, la fase de inicialización de los algoritmos resulta crítica y debe especificarse cuidadosamente.

Para el problema considerado, la información disponible viene dada por el vector de observaciones, $\mathbf{y} = [y[0],\ \ldots,\ y[N]]^T$, que constituye el conjunto incompleto de datos, y el parámetro que se desea estimar es la primera muestra de la secuencia, $x[0]$. Respecto al conjunto de datos ocultos, resulta natural considerar el itinerario asociado a la secuencia caótica, $\mathbf{s} = [s[0],\ \ldots,\ s[N-1]]^T$, puesto que su conocimiento facilita enormemente la estimación, como ya se ha visto en los capítulos 3 y 4. Por lo tanto, el

conjunto de datos completos es $\mathbf{z} = [\mathbf{y}^T,\ \mathbf{s}^T]^T$, y el estimador ML de $x[0]$ resulta:

$$\hat{x}_{\mathrm{ML}}[0] = \arg\operatorname*{máx}_{x[0]} \left\{ \mathrm{E}_{\mathbf{s}|\mathbf{y}}(\ln p(\mathbf{y}, \mathbf{s}; x[0])) - \mathrm{E}_{\mathbf{s}|\mathbf{y}}(\ln p(\mathbf{s}|\mathbf{y}; x[0])) \right\}.$$

El algoritmo E-M intenta encontrar $\hat{x}_{\mathrm{ML}}[0]$ buscando iterativamente el valor de $x[0]$ que maximiza

$$Q(x[0], \hat{x}_{\mathrm{EM}}^{k-1}[0]) = \sum_{i=1}^{P(N)} \ln p(\mathbf{y}, \mathbf{s}_i; x[0]) p(\mathbf{s}_i|\mathbf{y}; \hat{x}_{\mathrm{EM}}^{k-1}[0]), \tag{5.15}$$

donde $\hat{x}_{\mathrm{EM}}^{k-1}[0]$ es la estima E-M de $x[0]$ en la iteración $(k-1)$-ésima. Y puesto que, dada una condición inicial, $\hat{x}_{\mathrm{EM}}^{k-1}[0]$, la secuencia simbólica queda perfectamente fijada, $p(\mathbf{s}_i|\mathbf{y}; \hat{x}_{\mathrm{EM}}^{k-1}[0]) = \delta(\mathbf{s}_i - \hat{\mathbf{s}}_{\mathrm{EM}}^{k-1})$, siendo $\hat{\mathbf{s}}_{\mathrm{EM}}^{k-1}$ la secuencia simbólica de longitud N asociada a $\hat{x}_{\mathrm{EM}}^{k-1}[0]$. Desarrollando la FDP conjunta de las observaciones y el itinerario utilizando el teorema de Bayes, resulta sencillo demostrar que maximizar (5.15) es equivalente a minimizar

$$J(x[0], \hat{x}_{\mathrm{EM}}^{k-1}[0]) = \sum_{n=0}^{N} \left(y[n] - f_{\hat{\mathbf{s}}_{\mathrm{EM}}^{k-1}}^{n}(x[0]) \right)^2,$$

donde la dependencia con $\hat{x}_{\mathrm{EM}}^{k-1}[0]$ se introduce a través de su itinerario, $\hat{\mathbf{s}}_{\mathrm{EM}}^{k-1}$. Resulta evidente que esta minimización conduce al estimador ML local de $x[0]$ correspondiente a $\hat{\mathbf{s}}_{\mathrm{EM}}^{k-1}$, dado por el Algoritmo 3.3, en una sola iteración. En consecuencia, para garantizar la obtención del estimador ML global se debería repetir el proceso con $P(N)$ condiciones iniciales arbitrarias dentro de cada una de las $P(N)$ posibles regiones de la partición obtenida en función del itinerario. Por lo tanto, esta primera formulación del algoritmo E-M no ayuda en absoluto a resolver de manera eficiente el problema, aunque seleccionando $\hat{x}_{\mathrm{EM}}^{0}[0] = y_T[0]$, el estimador obtenido coincide siempre con el HC-ML(0) y tiende asintóticamente al estimador ML exacto.

Para intentar disminuir el coste computacional del estimador, se va a dividir el problema de estimación en dos partes siguiendo la línea de [Nelson1996]:

1. Utilizar el algoritmo E-M (o alguna de sus variantes) para obtener una estima de la secuencia simbólica, $\hat{\mathbf{s}}_{\mathrm{EM}}$.

2. Estimar una muestra de referencia (generalmente $x[0]$ ó $x[N]$) usando $\hat{\mathbf{s}}_{\mathrm{EM}}$ y generar la estima del resto de la secuencia caótica iterando hacia delante y/o hacia atrás mediante el Algoritmo 3.3.

En consecuencia, la discusión llevada a cabo en el resto de esta sección se puede concentrar en el desarrollo del algoritmo E-M para la estimación del itinerario. En este punto, la dificultad se encuentra en realizar la división óptima de la secuencia simbólica en parámetros y datos ocultos con el fin de minimizar el coste computacional. Supóngase que se toman K símbolos ($1 \leq K \leq N-1$) como parámetros, de modo que $\boldsymbol{\theta} = [s[n_1],\ \ldots,\ s[n_K]]^T$, y el resto como datos ocultos, $\boldsymbol{\phi} = [s[n_{K+1}],\ \ldots,\ s[n_N]]^T$,

con $n_i \in \{0,\ 1,\ \ldots,\ N-1\}$ y $n_i \neq n_j\ \forall\ i,j$ con $1 \leq i,j \leq N$. Entonces, la función que debe maximizar el algoritmo E-M es

$$Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}_{\mathrm{EM}}^{k-1}) = \sum_{\boldsymbol{\Phi} \in \Phi^{k-1}} \ln p(\mathbf{y}, \boldsymbol{\Phi}; \boldsymbol{\theta}) p(\boldsymbol{\Phi}|\mathbf{y}; \hat{\boldsymbol{\theta}}_{\mathrm{EM}}^{k-1}), \tag{5.16}$$

donde Φ^{k-1} denota el conjunto de todos los posibles vectores de datos ocultos, $\boldsymbol{\Phi}$, que dan lugar a itinerarios válidos con $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}_{\mathrm{EM}}^{k-1}$. Alternativamente, considerando los parámetros a estimar como variables aleatorias y planteando la maximización directa de $p(\mathbf{y}|\mathbf{s}) = p(\mathbf{y}|\boldsymbol{\Phi}, \boldsymbol{\theta})$ en lugar de $p(\mathbf{y}, \boldsymbol{\Phi}|\boldsymbol{\theta})$, la función a maximizar por el algoritmo E-M se puede expresar como [Panta2002]:

$$Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}_{\mathrm{EM}}^{k-1}) = \sum_{\boldsymbol{\Phi} \in \Phi^{k-1}} \ln p(\mathbf{y}|\boldsymbol{\Phi}, \boldsymbol{\theta}) p(\mathbf{y}|\boldsymbol{\Phi}, \hat{\boldsymbol{\theta}}_{\mathrm{EM}}^{k-1}) p(\boldsymbol{\Phi}, \hat{\boldsymbol{\theta}}_{\mathrm{EM}}^{k-1}). \tag{5.17}$$

Esta última ecuación, que va a ser la usada en lo sucesivo, presenta la ventaja con respecto a (5.16) de que todas las FDPs que intervienen en la misma ya han sido calculadas previamente o pueden obtenerse con facilidad.

Para evaluar el coste computacional del algoritmo E-M, en primer lugar nótese que, aunque la cardinalidad de Φ^{k-1} puede variar en función de la división del itinerario realizada (esto es, de qué símbolos se consideren como parámetros y como datos ocultos) y de la iteración (es decir, del valor actual de $\hat{\boldsymbol{\theta}}_{\mathrm{EM}}^{k-1}$), siempre va a ser del orden del número de secuencias válidas de longitud $N-K$, $P(N-K)$. En consecuencia, la etapa E del algoritmo presenta un coste computacional $\Theta(P(N-K))$ por iteración. Respecto a la etapa M, requiere explorar un conjunto de K símbolos para encontrar el máximo, de modo que su coste es $\Theta(P(K))$ por iteración. Suponiendo que se realizan N_i iteraciones, el coste computacional total del algoritmo E-M resulta

$$C_3(N) = \Theta(N_i(P(K) + P(N-K))) \rightarrow \Theta(e^{h_0 K} + e^{h_0(N-K)}),$$

donde se ha eliminado la dependencia con N_i, ya que en general el número de iteraciones va a ser pequeño. Aunque el número de posibilidades a la hora de dividir la secuencia simbólica, $\sum_{i=1}^{N-2} \binom{N-1}{i}$, puede ser muy elevado, a continuación se van a analizar tres casos particulares de especial interés:

1. Considerar todos los símbolos como datos ocultos con excepción de uno, el n-ésimo, que se toma como el parámetro a estimar. En este caso, ya explorado en [Panta2002], $K = 1$ y la función de coste que se debe maximizar es

$$Q(s[n], \hat{s}_{\mathrm{EM}}^{k-1}[n]) = \sum_{\tilde{\mathbf{s}}_n \in \tilde{S}_n^{k-1}} \ln p(\mathbf{y}|\tilde{\mathbf{s}}_n, s[n]) p(\mathbf{y}|\tilde{\mathbf{s}}_n, \hat{s}_{\mathrm{EM}}^{k-1}[n]) p(\tilde{\mathbf{s}}_n, \hat{s}_{\mathrm{EM}}^{k-1}[n]). \tag{5.18}$$

   donde el vector de parámetros es $\theta = s[n]$, el de datos ocultos es $\boldsymbol{\Phi} = \tilde{\mathbf{s}}_n = \mathbf{s}_{0:n-1,n+1:N-1} = [s[0],\ \ldots,\ s[n-1],\ s[n+1],\ \ldots,\ s[N-1]]^T$ y $\tilde{S}_n^{k-1}$ denota el

conjunto de todos los posibles vectores $\tilde{\mathbf{s}}_n$ que dan lugar a secuencias simbólicas válidas con $s[n] = \hat{s}_{\mathrm{EM}}^{k-1}[n]$. Esta formulación simplifica enormemente la etapa M, ya que sólo hay que probar como mucho M símbolos, pero a costa de introducir una gran complejidad en la etapa E, que requiere un sumatorio de hasta M^{N-1} términos, por lo que no resulta útil en la práctica.

2. Considerar todos los símbolos como parámetros con excepción de uno, el n-ésimo, que se toma como dato oculto. Esto es, se trata de la formulación complementaria a la anterior, en la que $K = N - 1$, y da lugar a la siguiente función de coste a maximizar

$$Q(\tilde{\mathbf{s}}_n, \hat{\tilde{\mathbf{s}}}_n^{k-1}(\mathrm{EM})) = \sum_{s[n] \in S^{k-1}} \ln p(\mathbf{y}|s[n], \tilde{\mathbf{s}}_n) p(\mathbf{y}|s[n], \hat{\tilde{\mathbf{s}}}_n^{k-1}(\mathrm{EM})) p(s[n], \hat{\tilde{\mathbf{s}}}_n^{k-1}(\mathrm{EM})), \tag{5.19}$$

donde S^{k-1} denota el conjunto de posibles valores de $s[n]$ que generan itinerarios válidos con $\tilde{\mathbf{s}}_n = \hat{\tilde{\mathbf{s}}}_n^{k-1}(\mathrm{EM})$. Desafortunadamente esta formulación tampoco resulta interesante, ya que ahora se tiene el problema inverso al anterior: la etapa E resulta muy sencilla, sólo es necesario sumar como mucho M términos, pero la etapa M es costosa, requiriendo la maximización conjunta de $N - 1$ símbolos.

3. Teniendo en cuenta el crecimiento exponencial asintótico del número de regiones, resulta sencillo demostrar que la elección óptima si se pretende minimizar el coste computacional es $K = N/2$ si N es par y $K = (N \pm 1)/2$ si N es impar. En este caso el coste es $\Theta(2P(N/2))$ por iteración.

Así pues, para finalizar esta sección, nótese que se han obtenido dos conclusiones muy importantes:

1. Se observa el compromiso habitual del algoritmo E-M: cuanto mayor sea la información adicional introducida a través de los datos ocultos más se simplifica la etapa M, pero a costa de complicar la etapa E, y viceversa, poca información adicional da lugar a una etapa E sencilla y una etapa M muy costosa.

2. Aunque el algoritmo E-M permite reducir en cierta medida el coste computacional con respecto a la minimización directa de la función de verosimilitud, no se consigue evitar su crecimiento exponencial con la longitud de la secuencia. La diferencia con el problema de la detección multiusuario [Nelson1996], donde el coste computacional sí se reduce, estriba en que en el caso de la detección multiusuario existe una dependencia lineal de la salida con la secuencia de bits, mientras que en este caso su dependencia con el itinerario es fuertemente no lineal.

No obstante, a pesar de que el algoritmo E-M no conduce a ningún método de estimación computacionalmente eficiente por sí mismo, la formulación desarrollada para el mismo y las consideraciones realizadas en esta sección van a resultar muy útiles para plantear técnicas computacionalmente eficientes basadas en una variante del E-M: la familia de algoritmos SAGE.

5.5.3. Estimación de Señales Caóticas con Algoritmos SAGE

La familia de algoritmos SAGE se basa en el mismo principio que el algoritmo E-M, de modo que se puede usar la misma formulación. La principal diferencia entre ambos estriba en que el SAGE no actualiza todos los parámetros en cada iteración, sino únicamente un subconjunto de los mismos. En consecuencia, el SAGE presenta una gran ventaja frente al E-M: puede implementarse sin datos ocultos, seleccionando sucesivamente diferentes variables del vector de parámetros [Nelson1996]. En este caso la etapa E no resulta necesaria y los parámetros que no se actualicen en una cierta iteración simplemente conservan el valor de la iteración anterior. Esta es la metodología que se va a seguir en esta sección, con dos variantes: considerando únicamente los símbolos del itinerario como parámetros (y estimando posteriormente la secuencia caótica mediante el Algoritmo 3.3), e incluyendo adicionalmente la condición inicial, $x[0]$ (y obteniéndose el resto de la secuencia iterando).

Para el primer algoritmo, SAGE1, se utiliza $\boldsymbol{\theta} = \mathbf{s}$, seleccionándose en la etapa D de la iteración k-ésima ($1 \leq k \leq N_i$) únicamente el símbolo l-ésimo, con $l = ((k-1) \mod N)$, donde ($a \mod b$) indica la operación módulo, que devuelve el resto entero del cociente a/b. Puesto que no hay datos ocultos, no es necesaria la etapa E, y la FDP que se debe maximizar en la etapa M es directamente la de las observaciones condicionada por los parámetros: $p(\mathbf{y}|s[l], \hat{\tilde{\mathbf{s}}}_l^{k-1})$. Esta FDP se puede obtener fácilmente a partir de $p(\mathbf{y}, x[0]|s[l], \hat{\tilde{\mathbf{s}}}_l^{k-1})$, ya conocida, integrando con respecto a $x[0]$:

$$\begin{aligned} p(\mathbf{y}|s[l], \hat{\tilde{\mathbf{s}}}_l^{k-1}) &= \int_X p(\mathbf{y}, x[0]|s[l], \hat{\tilde{\mathbf{s}}}_l^{k-1}) dx[0] \\ &= \sum_{i=1}^{M} \int_X p(\mathbf{y}, x[0]|s_i[l], \hat{\tilde{\mathbf{s}}}_l^{k-1}) \delta(s[l] - s_i[l]) dx[0], \end{aligned} \tag{5.20}$$

donde X denota el espacio de fases del mapa caótico y $s[l]$ es el parámetro con respecto al cual se desea maximizar (5.20). Para desarrollar el algoritmo SAGE1 es necesario obtener una expresión analítica de la FDP conjunta de las observaciones y la condición inicial dado el itinerario, que es la que aparece dentro de la integral en (5.20), y que se puede expresar como

$$p(\mathbf{y}, x[0]|\hat{\mathbf{s}}_i^{k-1}) = p(\mathbf{y}|x[0], \hat{\mathbf{s}}_i^{k-1}) p(x[0]|\hat{\mathbf{s}}_i^{k-1}), \tag{5.21}$$

donde $\hat{\mathbf{s}}_i^{k-1} = [\hat{s}^{k-1}[0],\ \ldots,\ \hat{s}^{k-1}[l-1],\ s_i[l],\ \hat{s}^{k-1}[l+1],\ \ldots,\ \hat{s}^{k-1}[N-1]]^T$. A continuación, nótese que $p(\mathbf{y}|x[0], \hat{\mathbf{s}}_i^{k-1})$ es simplemente una Gaussiana $(N+1)$-dimensional con media $\boldsymbol{\mu} = [x[0],\ f_{\hat{s}_i^{k-1}[1]}(x[0]),\ \ldots,\ f^N_{\hat{\mathbf{s}}_i^{k-1}}(x[0])]^T$ y matriz de covarianza $\mathbf{C} = \sigma^2 \mathbf{I}$,

$$p(\mathbf{y}|x[0], \hat{\mathbf{s}}_i^{k-1}) = \frac{1}{(2\pi\sigma^2)^{(N+1)/2}} \exp\left(-\frac{1}{2\sigma^2} J(x[0], \hat{\mathbf{s}}_i^{k-1}) \right), \tag{5.22}$$

siendo $J(x[0], \hat{\mathbf{s}}_i^{k-1})$ la función de coste habitual utilizada en los capítulos 3 y 4, que, a la vista de los resultados de las secciones 4.3.1.2 y 4.3.1.3, se puede poner como

$$\frac{1}{2\sigma^2} J(x[0], \hat{\mathbf{s}}_i^{k-1}) = \frac{(x[0] - \mu_i)^2}{2\sigma_i^2} + \frac{c_i^2 - \mu_i^2}{2\sigma_i^2},$$

gracias a lo cual (5.22) resulta

$$p(\mathbf{y}|x[0],\hat{\mathbf{s}}_i^{k-1}) = \frac{1}{(2\pi\sigma^2)^{(N+1)/2}} \cdot \frac{q_i}{p_i} \exp\left(-\frac{(x[0]-\mu_i)^2}{2\sigma_i^2}\right), \tag{5.23}$$

donde μ_i, σ_i^2, c_i^2 y q_i están definidas como en la Sección 4.3.1.2, y p_i es el valor de la FDP a priori de $x[0]$ dentro de la región correspondiente al itinerario i-ésimo. Respecto a la segunda FDP, la de $x[0]$ dada la secuencia simbólica i-ésima, utilizando la aproximación PWC para la FDP invariante introducida en el Capítulo 4 (exacta para mapas PWL de Markov), se trata de una FDP uniforme dentro de la región delimitada por dicha secuencia:

$$p(x[0]|\hat{\mathbf{s}}_i^{k-1}) = \frac{1}{\Delta_i}\chi_i(x[0]). \tag{5.24}$$

Sustituyendo (5.23) y (5.24) en (5.21) se encuentra que

$$p(\mathbf{y},x[0]|\hat{\mathbf{s}}_i^{k-1}) = \frac{1}{(2\pi\sigma^2)^{(N+1)/2}} \frac{q_i}{p_i\Delta_i} \exp\left(-\frac{(x[0]-\mu_i)^2}{2\sigma_i^2}\right) \chi_{R_i}(x[0]),$$

e introduciendo esta expresión en (5.20):

$$p(\mathbf{y}|\hat{\mathbf{s}}^{k-1}) = \frac{1}{(2\pi\sigma^2)^{(N+1)/2}} \sum_{i=1}^{M} \frac{q_i}{p_i\Delta_i} \delta(s[l]-s_i[l]) \int_{R_i} \exp\left(-\frac{(x[0]-\mu_i)^2}{2\sigma_i^2}\right) dx[0]. \tag{5.25}$$

Por último, recordando de nuevo los resultados de la Sección 4.3.1.2 resulta evidente que la integral que aparece finalmente en (5.25) vale $(K_i q_i)^{-1}$, y teniendo en cuenta además que $\Pr(x \in R_i) = p_i\Delta_i$ y eliminando aquellos términos que no dependen de $s[l]$, la expresión para la actualización del símbolo l-ésimo en la etapa M es

$$\begin{aligned} \hat{s}^k[l] &= \underset{s_i[l]}{\arg\max} \left\{[\Pr(x \in R_i)K_i]^{-1}\right\} \\ &= \underset{s_i[l]}{\arg\max} \left\{ \frac{\sigma_i}{\Delta_i} \exp\left(-\frac{c_i^2-\mu_i^2}{2\sigma_i^2}\right) \left[\operatorname{erf}\left(\frac{\kappa_i-\mu_i}{\sqrt{2\sigma_i^2}}\right) - \operatorname{erf}\left(\frac{\eta_i-\mu_i}{\sqrt{2\sigma_i^2}}\right)\right]\right\}, \end{aligned} \tag{5.26}$$

donde se ha usado (4.25) para obtener la expresión final. El Algoritmo 5.3 muestra el estimador SAGE1 completo: inicialización, estimación iterativa del itinerario (etapas D, E y M), y estimación de la secuencia caótica.

Una segunda posibilidad, ya considerada en [Panta2002], consiste en incorporar la condición inicial al vector de parámetros: $\boldsymbol{\theta} = [x[0],\ \mathbf{s}^T]^T$. En este caso, en cada iteración se actualizan $x[0]$ y un símbolo del itinerario elegido del mismo modo que en el algoritmo SAGE1. Puesto que en esta ocasión tampoco existen datos ocultos, de nuevo resulta innecesaria la etapa E, y la etapa M de este segundo algoritmo, SAGE2, busca maximizar (5.22) con respecto a $x[0]$ y $s[l]$, o equivalentemente minimizar la función de coste cuadrática habitual:

$$\left\{\hat{x}^k[0],\ \hat{s}^k[l]\right\} = \underset{x[0],s[l]}{\arg\min}\ J(x[0],s[l],\hat{\tilde{\mathbf{s}}}_l^{k-1}). \tag{5.27}$$

1. Se inicializa el vector de parámetros (la secuencia simbólica) al valor del itinerario de la secuencia truncada. Es decir, $\hat{s}^0[k] = i \Leftrightarrow y_T[k] \in E_i$, con $0 \leq k \leq N-1$, y siendo $y_T[k] = \text{máx}(e_0, \text{mín}(e_M, y[n]))$.
2. Mientras el algoritmo no haya convergido, para la iteración k-ésima ($1 \leq k \leq N_i = n_v \cdot N$):
   2.1. Etapa D: Seleccionar como parámetro a actualizar el símbolo l-ésimo, con $l = ((k-1) \mod N)$.
   2.2. Etapa E: No es necesaria en esta ocasión, ya que no existen datos ocultos.
   2.3. Etapa M: Actualizar el símbolo l-ésimo mediante la ecuación (5.26), dejando el resto de símbolos sin modificar: $\hat{s}^k[i] = \hat{s}^{k-1}[i]$, para $0 \leq i \leq N-1$ con $i \neq l$.
3. Se toma como estima ML potencial del itinerario el valor de la última iteración, $\hat{\mathbf{s}}^{N_i}$, y se obtiene el estimador ML de $x[0]$ mediante el Algoritmo 3.3, utilizando la secuencia simbólica dada por $\hat{\mathbf{s}}^{N_i}$.

Algoritmo 5.3: Algoritmo SAGE1 para la estimación de señales caóticas.

La implementación del SAGE2, que proporciona una nueva estima de $x[0]$ en cada iteración, se muestra en el Algoritmo 5.4. La principal diferencia entre los algoritmos SAGE1 y SAGE2 radica en que el primero busca el valor más probable del itinerario (y únicamente estima $x[0]$ en la iteración final), mientras que el segundo intenta encontrar directamente el valor más verosímil de $x[0]$.

Merece la pena señalar que teóricamente se podría conseguir una mejora en el rendimiento del algoritmo modelando los símbolos como variables continuas (es decir, usando símbolos "blandos" en lugar de "duros" como se ha hecho en esta sección). Esto daría lugar a itinerarios fraccionales, que indicarían la fiabilidad de la estima de cada símbolo, aunque también complicaría los algoritmos. Obviamente, en la última iteración estos símbolos se pasarían a través de un umbral duro para obtener las estimas finales. Sin embargo, este algoritmo no se ha llegado a implementar debido al buen rendimiento observado de los algoritmos SAGE1 y SAGE2, y se menciona aquí simplemente como una posible línea futura de investigación.

Para finalizar, nótese que los algoritmos E-M y SAGE también se pueden modificar fácilmente para obtener estimas de máxima probabilidad a posteriori (MAP) en lugar de estimas de máxima verosimilitud (ML) simplemente incluyendo la información de la probabilidad a priori en la función de coste. No obstante, dicha alternativa no se ha explorado en la presente Tesis, siendo otra posible línea futura de investigación.

5.6. Algoritmo de Viterbi

En esta sección se proporciona una introducción al algoritmo de Viterbi, seguida de su aplicación a la estimación de secuencias caóticas. El algoritmo de decodificación

1. Se inicializa el vector de parámetros (la secuencia simbólica, $\mathbf{s}$, y la condición inicial, $x[0]$) al valor del itinerario de la secuencia truncada y al estimador ML de $x[0]$ para dicha secuencia respectivamente. Es decir, $\hat{s}^0[k] = i \Leftrightarrow y_T[k] \in E_i$, con $0 \leq k \leq N-1$ e $y_T[k] = \text{máx}(e_0, \text{mín}(e_M, y[n]))$, y $\hat{x}^0[0] = \hat{x}^r_{\text{ML}}[0]$, asumiendo que la región delimitada por $\hat{\mathbf{s}}^0$ es la r-ésima.

2. Mientras el algoritmo no haya convergido, para la iteración k-ésima ($1 \leq k \leq N_i = n_v \cdot N$):

   2.1. Etapa D: Seleccionar como parámetro a actualizar la condición inicial, $x[0]$, y el símbolo l-ésimo, $s[l]$, con $l = ((k-1) \mod N)$.

   2.2. Etapa E: Tampoco es necesaria en esta ocasión, ya que no existen datos ocultos.

   2.3. Etapa M: Actualizar $x[0]$ y $s[l]$ mediante la ecuación (5.27), dejando el resto de símbolos sin modificar: $\hat{s}^k[i] = \hat{s}^{k-1}[i]$ para $0 \leq i \leq N-1$ con $i \neq l$. De manera más explícita, la ecuación de actualización para el itinerario es:

$$\hat{s}^k[i] = \begin{cases} s_r[l], & i = l; \\ \hat{s}^{k-1}[i], & i \neq l; \end{cases}$$

   con $r = \arg\min_i J(\hat{x}^i_{\text{ML}}[0], s_i[l], \hat{\tilde{s}}^{k-1}_l)$, siendo $\hat{x}^i_{\text{ML}}[0]$ el estimador ML local correspondiente al itinerario i-ésimo de entre los M posibles. Asimismo,

$$\hat{x}^k[0] = \hat{x}^r_{\text{ML}}[0].$$

3. Se toma como estima ML potencial de $x[0]$ y $\mathbf{s}$ el valor de la última iteración: $\hat{x}^{N_i}[0]$ y $\hat{\mathbf{s}}^{N_i}$. El resto de la secuencia caótica se obtiene simplemente iterando.

Algoritmo 5.4: Algoritmo SAGE2 para la estimación de señales caóticas.

de Viterbi (VA) fue desarrollado originalmente por Andrew Viterbi en 1967 como un método óptimo para la decodificación de códigos convolucionales [Viter1967]. Posteriormente, Omura y Forney mostraron que el algoritmo también se puede aplicar a la igualación y detección de señales de comunicaciones digitales en presencia de interferencia entre símbolos (ISI) [Omura1971, Forney1972]. Además, Forney demostró que el algoritmo de Viterbi es en realidad el estimador ML de la secuencia de símbolos transmitidos [Forney1972]. En consecuencia es siempre óptimo, y se puede modificar fácilmente para obtener el estimador MAP de la secuencia en lugar del ML [Forney1973]. Desde entonces, el algoritmo de Viterbi se ha aplicado en problemas muy diferentes: la estimación e igualación ciega de canal [Cao2002], el seguimiento de blancos con múltiples modelos [Ho2003], la estimación de la frecuencia instantánea de una señal [Stanko2003], el reconocimiento de voz [Yoma2003], etc.

El algoritmo de Viterbi en realidad es un método que permite hallar el camino óptimo (es decir, el de menor coste) a través de un grafo de una manera eficiente utilizando técnicas propias de *programación dinámica*. En comunicaciones digitales, el

VA posibilita la obtención del estimador ML o MAP de la secuencia de bits transmitidos con un coste computacional razonable siempre que el canal sea de memoria finita [Hayes1975, Sklar2003]. En este caso, el VA realiza de manera conjunta la igualación y la detección de la secuencia transmitida, y a menudo se conoce como igualador de Viterbi (VEQ). En la Sección 5.6.1 se describe brevemente su funcionamiento, y en la Sección 5.6.2 se muestra el modo de aplicarlo a la estimación de secuencias caóticas.

5.6.1. Introducción al Algoritmo de Viterbi y su Aplicación en Comunicaciones

Para la presentación del algoritmo de Viterbi se va a considerar el modelo más simple de un sistema de comunicaciones digitales sujeto a ISI posible: un esquema con modulación en la amplitud de los pulsos (sistema PAM). El diagrama de bloques del equivalente discreto en banda base de este sistema se muestra en la Figura 5.1, incluyendo el transmisor (modulador), el canal, y el receptor (demodulador y detector).

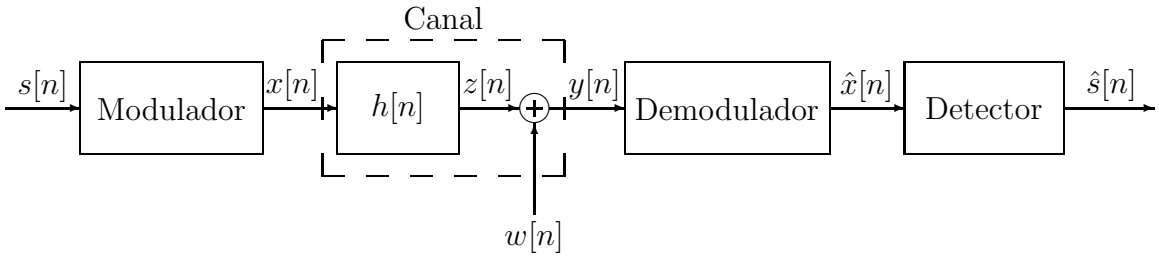


Figura 5.1: Diagrama de bloques del equivalente discreto en banda base de un sistema de comunicaciones digitales PAM.

Suponiendo que se desean transmitir $N+1$ símbolos, $\mathbf{s} = [s[0],\ \ldots,\ s[N]]^T$, y que el canal está compuesto por un filtro FIR de longitud L, $\mathbf{h} = [h[0],\ \ldots,\ h[L-1]]^T$, más AWGN, $\mathbf{w} = [w[0],\ \ldots,\ w[N+L-1]]^T$ con $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$, la señal a la salida del mismo se puede expresar como $\mathbf{y} = [y[0],\ \ldots,\ y[N+L-1]]^T$, con

$$y[n] = z[n] + w[n] = x[n] * h[n] + w[n] = \sum_{m=0}^{L-1} h[m]x[n-m] + w[n], \tag{5.28}$$

donde '*' denota la operación de convolución, $z[n] = x[n] * h[n]$ es la secuencia transmitida distorsionada por la ISI pero sin ruido, y $\mathbf{x} = [x[0],\ \ldots,\ x[N]]^T$ es el vector con las amplitudes de los $N+1$ pulsos transmitidos. Si se consideran símbolos M-arios ($s[n] \in \{s_1,\ \ldots,\ s_M\}$) e independientes, entonces $x[n]$ únicamente puede tomar M valores distintos (esto es, $x[n] \in \{x_1,\ \ldots,\ x_M\}$), que se encuentran unívocamente relacionados con cada posible valor de $s[n]$: $x_i = T(s_i)$. Por ejemplo, en el caso de una modulación M-PAM bipolar, habitualmente $s[n] \in \{0,\ \ldots,\ M-1\}$, y $x[n]$ se construye a partir de $s[n]$ mediante una transformación afín [Proak1998]: $x[n] = (2s[n] - M + 1)A$.

El objetivo último del receptor es obtener una estima de los símbolos transmitidos, $\hat{s}[n]$, que minimice la probabilidad de haber cometido un error con respecto al símbolo real, $s[n]$. Cuando todos los símbolos son equiprobables, esta estima viene dada por el estimador ML de la secuencia de símbolos (MLSE), que se obtiene minimizando el error cuadrático medio entre el vector de observaciones, $\mathbf{y}$, y el vector con la señal filtrada pero sin ruido, $\mathbf{z} = [z[0], \ \ldots, \ z[N+L-1]]^T$. Cuando los símbolos no son equiprobables, una mejor estima viene dada por el estimador MAP de la secuencia de símbolos (MAPSE), que se obtiene introduciendo en la función de coste a minimizar la probabilidad de ocurrencia de cada una de las M^N secuencias simbólicas.

La aproximación de "fuerza bruta" o "búsqueda exhaustiva" al problema requeriría estimar la respuesta al impulso del canal, $\hat{\mathbf{h}}$, probar las M^{N+1} posibles señales transmitidas (distorsionadas usando $\hat{\mathbf{h}}$ para tener en cuenta el efecto de la ISI), y seleccionar aquella que minimice la función de coste correspondiente. Desafortunadamente, el crecimiento exponencial del coste computacional con la longitud de la secuencia transmitida imposibilita el uso de este método para valores de N medios y altos.

Para construir un estimador eficiente de $\mathbf{s}$ resulta fundamental darse cuenta de que el sistema se comporta como un *proceso de Markov*: su salida en un instante determinado sólo depende del estado en el que se encuentre y de la entrada actual. El *estado* del sistema en el instante n-ésimo está compuesto por todos aquellos símbolos disponibles hasta ese momento que influyen en la siguiente salida del canal, $z[n+1]$. En consecuencia, puesto que $z[n+1]$ es únicamente función de $\mathbf{x}_{l:m}$, con $l = n-(L-2)$ y $m = n$ excepto durante las fases de inicialización y finalización del algoritmo, entonces el estado del sistema en general viene dado por $\mathbf{s}_{n-(L-2):n}$. Además, como cada símbolo únicamente puede tomar M valores diferentes, el sistema se puede caracterizar como una *máquina de estados finitos* (FSM) con M^{L-1} estados, M ramas salientes de cada estado actual, y otras tantas entrantes en los estados próximos, lo que da lugar a M^L posibles transiciones entre estados en cada iteración.

Este carácter markoviano del sistema, producto de la memoria finita del canal, junto con el número finito de posibles salidas, es lo que aprovecha el algoritmo de Viterbi para resolver el problema de manera exacta y eficiente en tres etapas [Sklar2003]:

1. Construcción de un diagrama de "trellis" (enrejado en su traducción literal), que muestra los M^{L-1} estados en el eje vertical, los $N+L$ instantes temporales en el eje horizontal, y las transiciones entre estados válidas mediante uniones en el diagrama (ramas).

2. Recorrido del "trellis" de izquierda a derecha (esto es, desde $n = 0$ hasta $n = N+L-1$), calculando el coste de cada transición y de cada estado. Para cada instante de tiempo y estado únicamente se almacena el mejor camino que termina en el mismo para su posible uso futuro.

3. Una vez llegado al extremo derecho del "trellis", recorrido del mismo de derecha a izquierda (es decir, desde $n = N+L-1$ hasta $n = 0$), partiendo del estado con

menor coste y determinando los símbolos transmitidos a partir del mejor camino almacenado en la etapa anterior.

El coste computacional del estimador óptimo utilizando el algoritmo de Viterbi es $\Theta(N \cdot M^L)$, frente a $\Theta(M^{N+1})$ de la búsqueda exhaustiva. Puesto que generalmente L es mucho menor que N, esto supone una importante reducción del coste computacional, que usando el VA crece linealmente con N en lugar de exponencialmente. Sin embargo, aunque el VA es ampliamente utilizado en la práctica, presenta dos inconvenientes importantes: la necesidad de disponer de toda la secuencia recibida para decodificar, y un coste computacional que crece exponencialmente con la longitud del canal, L. El primer problema no resulta relevante para la Tesis, de modo que no se discuten sus posibles soluciones. Respecto al segundo, es debido a que el número de estados crece exponencialmente con la memoria del canal, pudiendo ser de hasta M^{L-1} cuando todos los símbolos son independientes.

Existen fundamentalmente dos clases de técnicas que tratan de resolver este problema: aquellas que realizan la búsqueda en un "trellis" reducido, y las que utilizan el "trellis" completo pero únicamente tienen en cuenta un número reducido de caminos. Entre las primeras, la opción más sencilla consiste en truncar directamente la respuesta al impulso del canal. Otras alternativas más sofisticadas consisten en combinar un igualador (lineal o no) con el VA [Quresh1973a, Lee1977, Duel1989] para reducir la longitud efectiva de $h[n]$, o agrupar estados de acuerdo a criterios de distancia, lo que se conoce como técnicas de detección/estimación con un número reducido de estados (RSSD o RSSE) [Eyubo1988, Ander1994, Matola1996]. Entre las segundas, las dos posibilidades básicas consisten en mantener solamente un cierto número fijo de camino (los mejores) en cada iteración [Jeline1971, Vermeu1974, Ander1975], o mantener un número variable (aunque acotado) e ir descartando aquellos cuyo coste exceda un determinado umbral [Foschi1977, Simmon1990, Zamiri1999b, Vikalo2003a]. Por último, también existen numerosos algoritmos que combinan ambas ideas [Wesolo1987, Zamiri2002].

5.6.2. Aplicación del Algoritmo de Viterbi a la Estimación de Señales Caóticas

Para un mapa caótico cualquiera, dada una condición final, $x[N]$, el resto de la señal caótica viene determinado de manera única por la secuencia simbólica, $\mathbf{s}$. Además, puesto que $s[n]$ únicamente puede tomar M valores distintos, el itinerario de la señal caótica desempeña un papel similar a la secuencia de símbolos a transmitir por una modulación M-PAM. En consecuencia, aunque van a existir algunas diferencias importantes, parece obvio que se puede construir un "trellis" en el que el itinerario de la señal caótica se utilice para determinar el estado del sistema. En la siguiente sección se muestran las dificultades a la hora de construir un "trellis" para la estimación de las señales caóticas, la manera de resolverlas y las causas del buen funcionamiento del algoritmo, a pesar de las simplificaciones realizadas. Posteriormente, en la Sección 5.6.2.2 se discute la métrica usada para las ramas y los nodos del "trellis", se representan los diferentes

"trellis" obtenidos para algunos de los mapas caóticos propuestos en el Capítulo 2, y se desarrolla el algoritmo implementado finalmente.

5.6.2.1. Dificultades y Compromisos para la Aplicación del VA

Aunque parezca evidente que se puede construir un "trellis" a partir de la señal caótica usando la secuencia simbólica para determinar el estado de la señal, a la hora de generar el "trellis" y aplicar el VA en la estimación de una secuencia caótica aparecen tres dificultades:

1. Si se genera la señal caótica iterando hacia delante de acuerdo con (2.8) como es habitual, cada posible itinerario tiene asociada una región a la que debe pertenecer $\hat{x}[0]$ que no se solapa con la de ningún otro itinerario. Esto es problemático, ya que implica la existencia de una única rama en el "trellis" saliendo de cada estado, de modo que al final el problema se reduciría a obtener una estima por cada posible secuencia simbólica y no se obtendría ninguna ventaja desde el punto de vista del coste computacional con respecto a la búsqueda exhaustiva.

2. Al contrario que una señal de comunicaciones digitales PAM, una señal caótica puede tomar infinitos valores dentro de su espacio de fases. En los problemas de estimación considerados en el Capítulo 3 y el Capítulo 4, en principio no se conoce ningún punto de la secuencia caótica. Por consiguiente, no se dispone de ningún punto de partida para calcular el equivalente de los $z[k]$ que requiere la obtención de la métrica de cada rama, y es necesario encontrar una muestra de referencia, $\hat{x}[n]$, de alguna manera para empezar la iteración del algoritmo.

3. Los sistemas caóticos considerados en esta Tesis se pueden ver como filtros IIR no lineales. Esto significa que tienen memoria infinita, y que por lo tanto no se puede aplicar el VA directamente, ya que sería necesario un número de estados igual a M^N, de modo que nuevamente el coste computacional del algoritmo sería el mismo que el de la búsqueda exhaustiva. En consecuencia, resulta necesario considerar alguna técnica de reducción del número de caminos y/o estados.

Aunque estos tres problemas son importantes, los dos primeros se pueden solucionar de forma óptima de manera sencilla, y el tercero de forma aproximada aunque satisfactoria, como se muestra a continuación:

1. El primer problema se soluciona fácilmente utilizando el método alternativo para generar una secuencia caótica dado por (2.9) consistente en iterar hacia atrás. De este modo, puesto que todos los mapas caóticos unidimensionales son no invertibles, en general va a existir más de una rama de salida de cada estado (y de llegada a los estados en la siguiente iteración). Además, para aquellos mapas en los que desde cada intervalo se puede alcanzar cualquier punto de su espacio de fases, todas las regiones de $x[n]$ son alcanzables desde cada región de $x[n+1]$, y por lo tanto la estructura del "trellis" va a ser idéntica a la de un sistema

de comunicaciones M-PAM sin codificación. En caso contrario, existen ramas inviables dentro del "trellis" que simplemente se eliminan, al igual que ocurre en una modulación M-PAM con algún tipo de codificación.

2. Para resolver el segundo problema se debe hallar un punto de partida para el VA. Con la formulación propuesta en el punto anterior esto equivale a encontrar al menos un valor de $\hat{x}[N]$. Una posibilidad sería considerar las M regiones del espacio de fases del mapa y utilizar como punto de partida algún punto dentro de cada región para cada uno de los M estados finales posibles, ya que no se conoce con certeza el estado final del sistema. Por ejemplo, una alternativa sencilla sería utilizar el punto intermedio de cada región, $\hat{x}_i[N] = (e_{i-1} + e_i)/2$ con $1 \leq i \leq M$, o incluso un punto aleatorio dentro las mismas. No obstante, se dispone de la observación N-ésima, $y[N]$, de modo que se puede plantear como punto de partida el estimador ML de $x[N]$ dentro de cada región dada esta única observación:

$$\hat{x}_i[N] = \begin{cases} e_{i-1}, & y[N] < e_{i-1}; \\ y[N], & y[N] \in E_i; \\ e_i, & y[N] > e_i; \end{cases} \tag{5.29}$$

con $i = 1, \ldots, M$. Se podrían considerar otras alternativas más sofisticadas, como utilizar el estimador HC-ML(0) de $x[N]$ dada toda la secuencia ruidosa. Sin embargo, (5.29) mejora la primera alternativa mencionada y proporciona muy buenos resultados. Además, cualquier otro estimador de $\hat{x}_i[N]$ incrementaría la complejidad del algoritmo de manera innecesaria, de modo que no se consideran otras alternativas en esta Tesis.

3. Puesto que el sistema caótico tiene memoria infinita, para usar el VA sería necesario construir un "trellis" con M^N estados. Esto es debido a que al iterar hacia atrás $s[N-k]$ interviene en todas las muestras de la secuencia caótica generadas para $n \leq k$ (esto es, $s[N-1]$ influye en todas las muestras de la secuencia). Por lo tanto, en la iteración n-ésima hacia atrás a partir de un cierto $\hat{x}_i[N]$ es posible tener hasta M^n valores distintos de $\hat{x}[N-n]$, no pudiéndose descartar ninguno si se quiere garantizar que se encuentra el camino óptimo. Esto significa que no se puede utilizar el VA de manera exacta, ya que no se obtendría ninguna ventaja con respecto a la "búsqueda exhaustiva" implementada en el Capítulo 3. Para solucionar este problema se va a acudir a alguna de las técnicas de truncamiento o reducción del "trellis". En concreto, se va a recurrir a la más sencilla posible, que consiste en utilizar únicamente los últimos r símbolos para determinar el estado del sistema (de modo que se van a requerir como mucho $R = M^r$ estados).

En general el número de estados que se van a utilizar en el "trellis" es muy reducido, tomándose con frecuencia $r = 1$, de modo que el número total de estados es simplemente M, y obteniéndose muy buenos resultados. Aunque puede parecer sorprendente que el VA funcione tan bien con un número tan reducido de estados, el motivo es la

utilización de la iteración hacia atrás a la hora de construir el "trellis". Así, mientras que es bien conocido que la iteración hacia delante de un mapa caótico provoca la separación en promedio de las trayectorias de puntos iniciales cercanos (sensibilidad a las condiciones iniciales), la iteración hacia atrás de dos muestras separadas utilizando el mismo itinerario en ambos casos tiende a acercarlas. Como resultado, si dos secuencias se obtienen mediante la iteración hacia atrás usando la misma secuencia simbólica, al cabo de unas pocas iteraciones su valor va a ser prácticamente idéntico, sin importar el valor de la condición final, $x[N]$, usada para generarlas. En consecuencia, en el "trellis" se puede descartar la peor de ellas, con la certeza de que no va a proporcionar valores de salida muy diferentes de la seleccionada.

De manera más formal, el motivo del buen funcionamiento del VA se encuentra en una generalización del razonamiento llevado a cabo en [Luengo2005a]. Considérense dos señales caóticas, $\mathbf{x}_1 = [x_1[0],\ \ldots,\ x_1[N]]^T$ y $\mathbf{x}_2 = [x_2[0],\ \ldots,\ x_2[N]]^T$, generadas iterando hacia atrás a partir de $x_1[N]$ y $x_2[N]$ usando dos secuencias simbólicas $\mathbf{s}_1$ y $\mathbf{s}_2$ tales que $s_1[k] = s_2[k] = s[k]$ para $0 \leq k \leq n-1$, pero que pueden tomar valores diferentes para $k = n,\ \ldots,\ N$. Se va a definir la distancia entre ambas señales caóticas en el instante n-ésimo como

$$d[n] = |x_1[n] - x_2[n]|. \tag{5.30}$$

Iterando hacia atrás, se tiene que

$$x_1[n-1] = f^{-1}_{s[n-1]}(x_1[n]) = \frac{x_1[n] - b_{s[n-1]}}{a_{s[n-1]}},$$

para la primera secuencia, y

$$x_2[n-1] = f^{-1}_{s[n-1]}(x_2[n]) = \frac{x_2[n] - b_{s[n-1]}}{a_{s[n-1]}},$$

para la segunda secuencia, de modo que su distancia es

$$d[n-1] = |x_2[n-1] - x_1[n-1]| = |a_{s[n-1]}|^{-1} d[n].$$

Este proceso se puede continuar de manera recursiva, y en general, utilizando la ecuación (3.19) obtenida en el Capítulo 3, es fácil demostrar que la distancia entre las dos secuencias en la muestra $(n-k)$-ésima es:

$$d[n-k] = \left| B^{1,k}_{\mathbf{s}_{n-k:n-1}} \right| d[n] = \prod_{l=1}^{k} |a_{s[n-l]}|^{-1} d[n], \tag{5.31}$$

con $k = 1,\ \ldots,\ n$.

En muchos mapas caóticos (como la familia de mapas de tienda de campaña o el mapa de Markov de la Figura 2.4), $|a_i| > 1$ para cualquier región, y en consecuencia $d[n-k] < d[n]$ para cualquier k y para todos los posibles itinerarios. Sin embargo,

existen otros mapas, como los dos presentados en la Figura 2.5, que tienen algún intervalo en el que su pendiente puede ser menor que uno. En este caso, pueden aparecer separaciones transitorias entre $\mathbf{x}_1$ y $\mathbf{x}_2$, pero en promedio dichas secuencias van a tender a acercarse. Para demostrar esto, considérese una secuencia caótica típica. La probabilidad de que una muestra de la señal caótica pertenezca a la región i-ésima viene dada por su FDP invariante, y por lo tanto en promedio se va a tener que

$$\left|B^{1,k}_{\mathbf{s}_{n-k:n-1}}\right| = \prod_{i=1}^{M} |a_i|^{-kp_i}, \tag{5.32}$$

donde $p_i = \Pr(x \in E_i)$. Por otro lado, el exponente de Lyapunov para un mapa PWL se puede expresar como [Beck1993]

$$l = \sum_{i=1}^{M} \Pr(x \in E_i) \ln |a_i| > 0, \tag{5.33}$$

donde la última condición es la que se tiene que cumplir precisamente para que el mapa sea caótico. Tomando la exponencial de (5.33) tenemos

$$\exp(l) = \exp\left(\sum_{i=1}^{M} p_i \ln |a_i|\right) = \prod_{i=1}^{M} \exp(p_i \ln |a_i|) = \prod_{i=1}^{M} |a_i|^{p_i} > 1. \tag{5.34}$$

Y manipulando ligeramente (5.32), resulta evidente que, en promedio,

$$\left|B^{1,k}_{\mathbf{s}_{n-k:n-1}}\right| = \left(\prod_{i=1}^{M} |a_i|^{p_i}\right)^{-k} = \exp(-kl) < 1.$$

En consecuencia, para cualquier mapa PWL caótico las secuencias $\mathbf{x}_1$ y $\mathbf{x}_2$ van a tender a acercarse tanto más cuanto mayor sea el exponente de Lyapunov del mapa, y por lo tanto tiene sentido utilizar un número reducido de estados. Por último, nótese que, aunque la demostración anterior se ha realizado para mapas PWL, es fácilmente extensible a cualquier mapa caótico.

5.6.2.2. "Trellis" para Diferentes Señales Caóticas y Estimación Usando el Algoritmo de Viterbi

En esta sección se muestra el "trellis" utilizado para diferentes mapas caóticos, así como la métrica que se va a usar en las distintas ramas y nodos. En primer lugar, recuérdese que se emplea un número reducido de estados $R = M^r$ determinado únicamente por los últimos r símbolos. A continuación, nótese que la función de coste del MLSE para $n+1$ símbolos se puede expresar como

$$\begin{aligned} J^{n+1}_{\mathrm{MLSE}}(\mathbf{s}_{N-(n+1):N-1}) &= \sum_{k=0}^{n+1} \left(y[N-k] - f_{\mathbf{s}_{N-k:N-1}}(x[N])\right)^2 \\ &= J^{n}_{\mathrm{MLSE}}(\mathbf{s}_{N-n:N-1}) + \left(y[N-(n+1)] - f^{-(n+1)}_{\mathbf{s}_{N-(n+1):N-1}}(x[N])\right)^2. \end{aligned}$$

Además, teniendo en cuenta que $f^{-(n+1)}_{\mathbf{s}_{N-(n+1):N-1}}(x[N]) = f^{-1}_{s[N-(n+1)]}(x[N-n])$, resulta evidente que el coste de tomar la rama j-ésima estando en el estado i-ésimo en la iteración n-ésima es

$$c^j_{\mathbf{s}_i}[n] = |y[N-(n+1)] - z^j_{\mathbf{s}_i}[n+1]|, \tag{5.35}$$

donde $z^j_{\mathbf{s}_i}[n+1]$ viene dada por

$$z^j_{\mathbf{s}_i}[n+1] = f_j^{-1}(\hat{x}_{\mathbf{s}_i}[N-n]),$$

siendo $\hat{x}_{\mathbf{s}_i}[N-n]$ la señal caótica correspondiente al estado i-ésimo en la iteración n-ésima. Nótese que en general $1 \leq j \leq M$, aunque para muchos mapas pueden existir transiciones inválidas, y $1 \leq i \leq R$, pudiendo aparecer de nuevo estados inalcanzables para algunos mapas, así como durante la inicialización y terminación del algoritmo. Para el MAPSE se puede plantear una ecuación similar a (5.35), simplemente añadiendo un nuevo término de ponderación que tiene en cuenta la probabilidad de que $s[n] = j$:

$$c^j_{\mathbf{s}_i}[n] = (y[N-(n+1)] - z^j_{\mathbf{s}_i}[n+1])^2 - 2\sigma^2 \ln \Pr(s[N-(n+1)] = j), \tag{5.36}$$

donde $\Pr(s[N-(n+1)] = j) = \Pr(x[N-(n+1)] \in E_j)$.

El coste de cada nodo o camino en el instante $(n+1)$-ésimo se obtiene del mismo modo que en el VA convencional: calculando el mínimo de los costes de todos los caminos que llegan a él, teniendo en cuenta los costes de los correspondientes estados anteriores y las ramas apropiadas. Para el estado j-ésimo en la iteración $(n+1)$-ésima este coste viene dado por

$$C_{\mathbf{s}_j}[n+1] = \min_{\mathbf{s}_i} \{C_{\mathbf{s}_i}[n] + c^j_{\mathbf{s}_i}[n]\},$$

para $0 \leq n \leq N-1$, siendo el coste de los M estados iniciales

$$C_{\mathbf{s}_j}[0] = |\hat{x}_{\mathbf{s}_j}[N] - y[N]|,$$

con $\hat{x}_{\mathbf{s}_j}[N]$ dado por (5.29). Nótese que, utilizando el VA reducido con $R = M^r$, esto significa que como mucho va a ser necesario considerar M^{r+1} ramas en cada iteración. Sin embargo, este número de ramas puede ser mucho más reducido para muchos mapas, resultando en general del orden de $P(r+1) \to e^{h_0(r+1)}$, el número de secuencias simbólicas de longitud $r+1$ válidas para el mapa caótico en cuestión.

En el caso en que $r = 1$ se puede simplificar la notación anterior para la métrica de las ramas y los estados [Luengo2005a, Luengo2005b, Luengo2005c]. La métrica de cada rama en la iteración n-ésima se puede expresar ahora para el MLSE como

$$c_{ij}[n] = |y[N-(n+1)] - f_j^{-1}(\hat{x}_i[n])|,$$

y para el MAPSE como

$$c_{ij}[n] = \left(y[N-(n+1)] - f_j^{-1}(\hat{x}_i[n])\right)^2 - 2\sigma^2 \ln \Pr(s[N-(n+1)] = j),$$

donde $1 \leq i, j \leq M$, y el coste de un nodo en la iteración $(n+1)$-ésima viene dado por

$$C_j[n+1] = \min_{1 \leq i \leq M} \{C_i[n] + c_{ij}[n]\},$$

para $0 \leq n \leq N-1$, siendo

$$C_j[0] = |\hat{x}_j[N] - y[N]|,$$

y teniendo aquellos estados y caminos inválidos asociado un coste infinito.

A continuación se van a mostrar algunos ejemplos de "trellis" para distintos mapas caóticos. El ejemplo más sencillo de diagrama de "trellis" de decodificación se obtiene cuando $r = 1$, $M = 2$, el mapa considerado es Markoviano, y todas las transiciones son posibles. Este es el caso por ejemplo del BSM, el TM y el S-TM con $\beta = 2$, el mapa logístico con $\lambda = 4$, el SK-TM o el BSK-TM. Para todos ellos el lazo básico del "trellis" es el que se muestra en la Figura 5.2, junto con los costes de cada rama y nodo. Nótese que en codificación los "trellis" tienen siempre un número de ramas que crece exponencialmente con la longitud de la secuencia. Sin embargo, puesto que no es necesario recurrir a ellos para generar las señales caóticas, no se muestran.

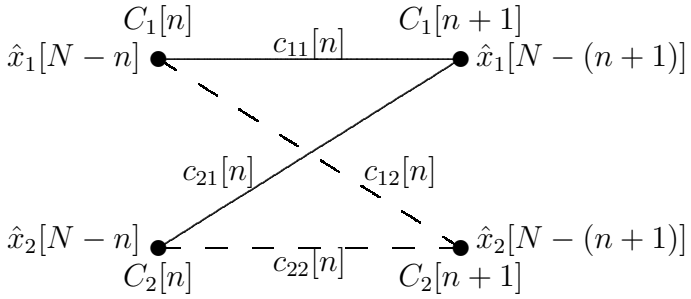


Figura 5.2: Lazo básico del VA en decodificación para un mapa de Markov en el que todas las transiciones entre estados son válidas, $r = 1$ y $M = 2$.

Para otros mapas, en general no todas las transiciones y estados son válidos. Como ejemplo, en la Figura 5.3 se muestra el lazo básico del "trellis" para el mapa de la Figura 2.4 y para los dos mapas de la Figura 2.5 cuando $r = 1$, y en la Figura 5.4 se presentan los mismos lazos cuando $r = 2$. Como se puede apreciar, cuando $r = 1$ el mapa denominado Markov1 presenta 4 estados y 12 transiciones, frente a los 12 estados y 34 transiciones cuando $r = 2$, creciendo rápidamente ambos conforme aumenta r. Para los otros dos mapas el crecimiento es mucho más suave: para el mapa Markov2, 2 estados y 3 transiciones con $r = 1$, y 3 estados con 5 transiciones en total para $r = 2$; para el mapa no markoviano, hasta 3 estados y 7 transiciones cuando $r = 1$, y hasta 7 estados y 17 transiciones cuando $r = 2$. Nótese que en todos ellos existen itinerarios inválidos que ya han sido eliminados de los "trellis" correspondientes.

En el caso de mapas de Markov (como los mapas Markov1 y Markov2, el TM y el S-TM con $\beta = 2$, o el SK-TM y BSK-TM) el diagrama de "trellis" se construye

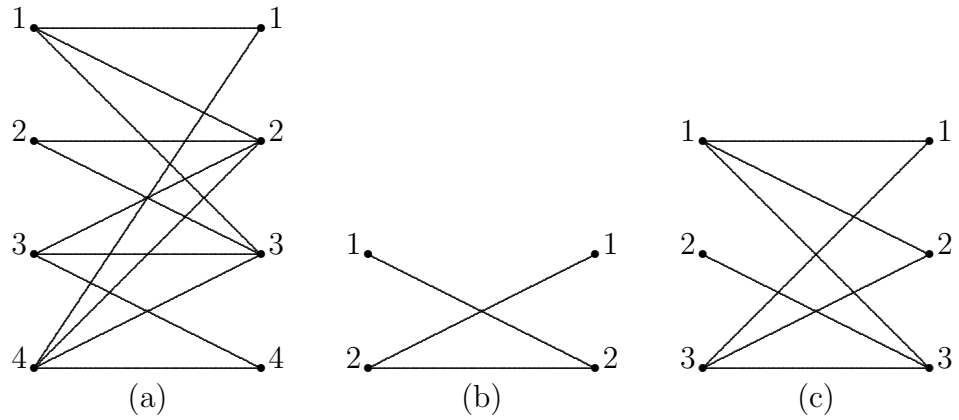


Figura 5.3: Lazo básico del VA para diversos mapas PWL cuando $r = 1$. (a) Lazo para el mapa Markov 1 de la Figura 2.4. (b) Lazo para el mapa Markov 2 de la Figura 2.5(a). (c) Lazo para el mapa PWL1 (no markoviano) de la Figura 2.5(b).

simplemente concatenando los lazos básicos correspondientes, excepto en las fases de inicialización y terminación del algoritmo, durante las cuales existe un número reducido de transiciones y estados válidos. En el caso de mapas no markovianos (como el mapa PWL1 de la Figura 2.5(b), el TM y el S-TM con $\beta < 2$, o el mapa logístico con $\lambda < 4$), la construcción del "trellis" no es tan sencilla. En estos casos no existe en general un lazo único, y el lazo real puede ir variando a lo largo del "trellis", existiendo estados y transiciones imposibles diferentes en cada iteración que se deben eliminar. Para este tipo de mapas, la mejor opción es, partiendo del lazo básico, ir comprobando según se va construyendo el "trellis" si sus transiciones y estados son válidos o no.

Para ilustrar esta situación, supóngase por ejemplo que se desea estimar la secuencia caótica $\mathbf{x} = [3/4,\ 59/60,\ 1/12,\ 2/5]^T$, generada usando el mapa PWL1 con un valor inicial $x[0] = 0{,}75$ y una secuencia simbólica $\mathbf{s} = [2,\ 3,\ 1,\ 1]^T$. Supóngase igualmente que la secuencia de observaciones disponibles es $\mathbf{y} = [0{,}7,\ 1{,}05,\ 0{,}24,\ 0{,}38]^T$, y que se va a usar $r = 1$. Para construir el "trellis" se deben calcular en primer lugar las muestras finales para los estados correspondientes a cada una de las regiones, tres en este caso. Puesto que $y[3] = 0{,}38$ pertenece a E_1, entonces $\hat{x}_1[3] = y[3] = 0{,}38$, y el coste de dicho estado inicial es $C_1[0] = 0$. Para las otras dos regiones se toma el punto más cercano a la observación (mínimo coste), obteniéndose $\hat{x}_2[3] = 0{,}5$ y $\hat{x}_3[3] = 0{,}8$, y un coste de los respectivos estados: $C_2[0] = 0{,}12$ y $C_3[0] = 0{,}42$.

A continuación se va a propagar la mejor estima de cada estado iterando hacia atrás siguiendo cada una de las tres ramas mostradas en el lazo básico de la Figura 5.3(c). Esto da lugar a un total de siete valores, que se muestran en la Tabla 5.2 agrupados según el estado de llegada. Como se puede apreciar en la tabla, dos de los siete valores se encuentran, bien fuera del espacio de fases del mapa, bien fuera de la región correspondiente al estado de llegada. En consecuencia estos valores se descartan, lo que equivale a podar las ramas correspondientes del "trellis". Además, las dos ramas eliminadas son las únicas que llegaban al estado 2 en $n = 2$, de modo que a efectos prácticos dicho estado se puede eliminar del "trellis", puesto que es imposible llegar a

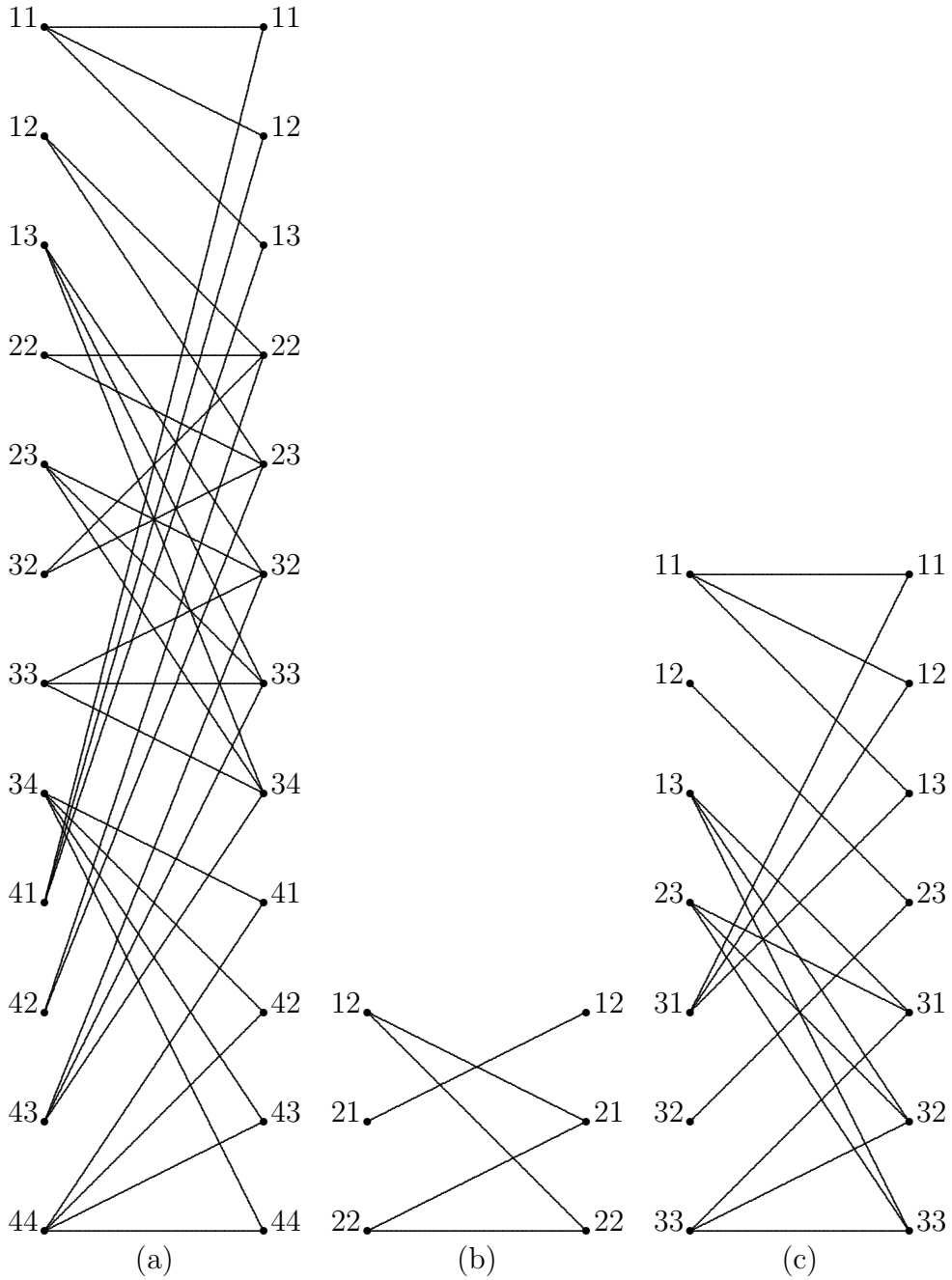


Figura 5.4: Lazo básico del VA para diversos mapas PWL cuando $r = 2$. (a) Lazo para el mapa Markov 1 de la Figura 2.4. (b) Lazo para el mapa Markov 2 de la Figura 2.5(a). (c) Lazo para el mapa PWL1 (no markoviano) de la Figura 2.5(b).

él en este ejemplo. En la Tabla 5.2 también se muestra el coste asociado a cada nodo, resaltándose en negrita el camino de menor coste para cada estado, que es el escogido por el algoritmo.

Se procede de igual modo para las dos siguientes iteraciones: generándose en cada caso todos los posibles candidatos, y descartándose aquellas ramas y estados inválidos.

$s[3]$	$s[2]$	$f_{s[2]}^{-1}(\hat{x}_{s[3]}[3])$	$C_{s[2]}(1)$
1	1	$f_1^{-1}(0{,}38) = 0{,}066 \in E_1$	**0,174**
3	1	$f_1^{-1}(0{,}8) = 0{,}416 \in E_1$	0,596
1	2	$f_2^{-1}(0{,}38) = -1{,}06 \notin I$	∞
3	2	$f_2^{-1}(0{,}8) = 0{,}2 \notin E_2$	∞
1	3	$f_3^{-1}(0{,}38) = 0{,}924 \in E_3$	**0,7**
2	3	$f_3^{-1}(0{,}5) = 0{,}9 \in E_3$	0,78
3	3	$f_3^{-1}(0{,}8) = 0{,}84 \in E_3$	1,02

Tabla 5.2: Conjunto de valores de la señal caótica posibles cuando $n = 2$ para el ejemplo de aplicación del VA propuesto con el mapa PWL1.

Al final, el "trellis" construido para este ejemplo se muestra en la Figura 5.5, donde se puede apreciar cómo en las dos últimas iteraciones el estado inválido es el 1. El valor de la señal de salida asociado a cada estado y su coste se muestran en la Tabla 5.3 y en la Tabla 5.4 respectivamente. Nótese cómo en este sencillo ejemplo $\hat{\mathbf{s}} = [2,\ 3,\ 1,\ 1]^T = \mathbf{s}$ y $\hat{\mathbf{x}} = [0{,}76,\ 0{,}9868,\ 0{,}066,\ 0{,}38]^T$, lo que implica que se ha reducido el error cuadrático medio de $7{,}972 \cdot 10^{-3}$ (lo que corresponde a una SNR de 17,25 dB) a $8{,}125 \cdot 10^{-4}$ (que equivale a una SNR de 27,18 dB).

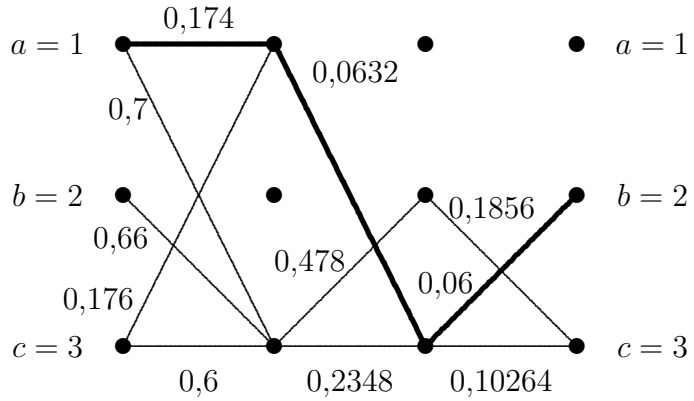


Figura 5.5: Diagrama de "trellis" de decodificación para un ejemplo de estimación de una señal caótica generada usando el mapa PWL1.

5.7. Resultados

En esta sección se presentan los principales resultados obtenidos para los distintos estimadores subóptimos propuestos a lo largo del capítulo agrupados en dos apartados. En primer lugar, en la Sección 5.7.1 se estudia el comportamiento de los dos estimadores sencillos desarrollados en las secciones 5.3 y 5.4: el FB-ML y el HC-ML(k). Y en segundo lugar, en la Sección 5.7.2 se analizan los estimadores más sofisticados de las secciones

n	3	2	1	0
$\hat{x}_1[n]$	**0,38**	**0,066**	N/A	N/A
$\hat{x}_2[n]$	0,5	N/A	0,572	**0,76**
$\hat{x}_3[n]$	0,8	0,924	**0,9868**	0,80264

Tabla 5.3: Valor de la señal caótica estimado en cada estado en función de la iteración del algoritmo para el ejemplo propuesto con el mapa PWL1. En negrita, el valor seleccionado en cada iteración.

n	0	1	2	3
$C_1[n]$	**0**	**0,174**	∞	∞
$C_2[n]$	0,12	∞	1,178	**0,2972**
$C_3[n]$	0,42	0,7	**0,2372**	0,33984

Tabla 5.4: Coste de cada estado en función de la iteración del algoritmo para el ejemplo propuesto con el mapa PWL1. En negrita, el valor seleccionado en cada iteración.

5.5 y 5.6, basados en los algoritmos E-M y de Viterbi, y que proporcionan mejores estimas, aunque a costa de incrementar ligeramente el coste computacional.

Respecto a las medidas de las prestaciones de los estimadores propuestos, se van a considerar las mismas que en el capítulo anterior: el MSE de la secuencia estimada y la tasa de aciertos en el itinerario. El MSE total de la secuencia estimada viene dado por (3.53), y resulta preferible en este caso al de $x[0]$ debido a que el bajo coste computacional de los estimadores planteados en este capítulo permite la estimación de secuencias "largas" (con $N = 29$ o $N = 99$ por ejemplo), para las que resulta más lógico evaluar la calidad del estimador de la secuencia completa que de una sola muestra. La tasa de aciertos en la secuencia simbólica estimada ya se utilizó en el Capítulo 4 para comparar el rendimiento de los estimadores Bayesianos con el del ML, mostrándose como un buen indicador de su calidad. Todos los estimadores subóptimos propuestos en este capítulo se basan en obtener primero una o varias estimas del itinerario de manera eficiente y posteriormente aplicar el estimador ML de la Sección 3.4.2. En consecuencia, dicha tasa de aciertos va a proporcionar una indicación muy precisa del rendimiento de cada estimador.

Por último, nótese que algunos de los estimadores subóptimos propuestos (FB-ML y HC-ML(k) con $k < N$) pueden dar lugar a itinerarios estimados inválidos, por lo que también se debe considerar la frecuencia con la que esto ocurre a la hora de evaluar su calidad. No obstante, en las simulaciones se ha observado que el porcentaje de estimas válidas es muy alto, excepto para valores de SNR muy bajos (por debajo de 20 dB), creciendo rápidamente hasta alcanzar prácticamente el 100 % para valores de SNR muy inferiores a los necesarios para que el MSE alcance el CRLB en el mejor de los casos o similares en el peor. En consecuencia, no se ha considerado interesante presentar gráficas de dicha tasa de estimas válidas en la siguiente sección.

5.7.1. Estimadores Sencillos (FB-ML y HC-ML)

En esta sección se comienza comparando el rendimiento de los estimadores FB-ML, HC-ML(k) y ML (HC-ML(N)) para secuencias cortas. En la Figura 5.6 se muestra el MSE total de la secuencia estimada, $\hat{\mathrm{M}}_T$(dB), para tres mapas unimodales distintos (S-TM, SK-TM y mapa Markov2 de la Figura 2.5(a)) con $N = 8$, así como para un mapa multimodal no markoviano, el mapa PWL1 de la Figura 2.5(b), para el que $M = 3$ y $N = 4$. En todos los casos el número de simulaciones de Monte Carlo llevadas a cabo para cada SNR y estimador es de 10.000.

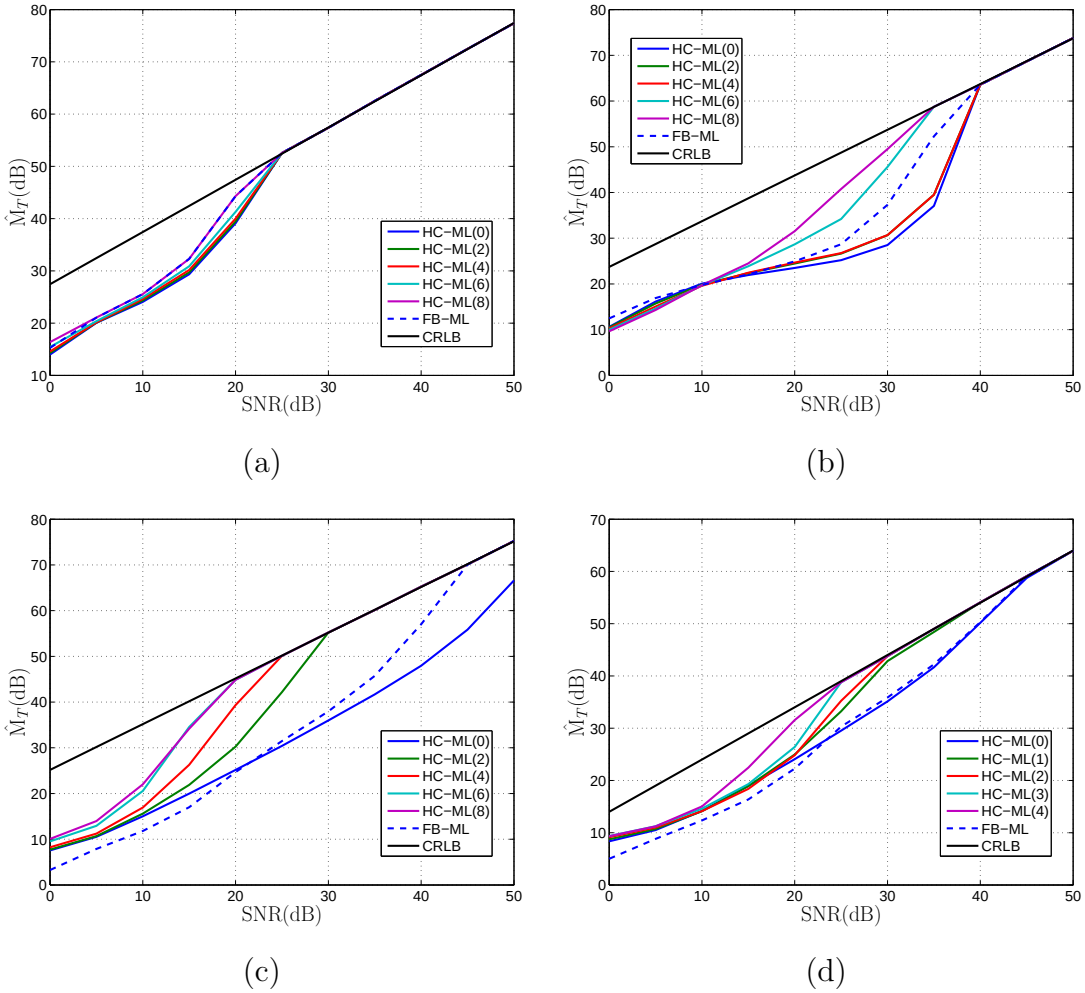


Figura 5.6: Comparación del rendimiento de los estimadores FB-ML y HC-ML(k) para secuencias generadas por distintos mapas caóticos. (a) S-TM con $\beta = 1{,}5$, $x[0] = 0{,}38316$ y $N = 8$. (b) SK-TM con $c = 0{,}1$, $x[0] = 0{,}2319$ y $N = 8$. (c) Mapa Markov2 de la Figura 2.5(a) con $c = 0{,}75$, $x[0] = 0{,}99582$ y $N = 8$. (d) Mapa PWL1 de la Figura 2.5(b) con $x[0] = 0{,}89734$ y $N = 4$.

Como se puede apreciar en la Figura 5.6(a), la ganancia del estimador ML (HC-ML(8)) con respecto al HC-ML(0) para el S-TM es muy pequeña, mientras que su coste computacional es 256 veces superior aproximadamente. Por otra parte, el FB-ML es óptimo para este mapa [Papado1993, Papado1995], por lo que su MSE coincide con el del HC-ML(8), y las diferencias (mínimas) observadas para valores muy bajos de SNR son achacables a errores numéricos.

Desafortunadamente, este buen comportamiento del FB-ML no se observa para el resto de los mapas caóticos, para los que su MSE se encuentra, bien a medio camino entre el del HC-ML(0) y el del ML, como en la Figura 5.6(b), bien cercano al del HC-ML(0), como en las Figuras 5.6(c) y (d). Otro tanto ocurre con el rendimiento del HC-ML(k): la ganancia en el MSE al aumentar k depende en gran medida del mapa, su parámetro de bifurcación y la condición inicial. Así, mientras que en la Figura 5.6(a) esta es mínima, en el resto de figuras el potencial de mejora del ML con respecto al FB-ML o al HC-ML(0) es enorme. Por ejemplo, en la Figura 5.6(c) el estimador ML alcanza el CRLB para SNR=20 dB, mientras que el FB-ML lo hace para una SNR de 45 dB, y el HC-ML(0) aún no lo ha alcanzado para SNR=50 dB.

Por último, nótese que en ocasiones existe un valor de k a partir del cual el comportamiento de los estimadores HC-ML(k) y ML es esencialmente el mismo para una SNR suficientemente elevada. Esto ocurre por ejemplo para $k = 6$ y SNR ≥ 15 dB en la Figura 5.6(c). El motivo es la existencia de muestras de la secuencia caótica muy alejadas del límite entre dos regiones, y en las que por tanto es muy poco probable que aparezca un error en el símbolo estimado. Por ejemplo, para la señal de la Figura 5.6(c), $x[1] = 0{,}0167$ y $x[4] = 0{,}0893$, siendo su distancia al límite con E_2, $d_1 = 0{,}7333$ y $d_4 = 0{,}6607$ respectivamente, muy superior al del resto de muestras de la secuencia.

A continuación se analiza la tasa de aciertos en el itinerario, mostrándose la misma para los cuatro ejemplos seleccionados de la Figura 5.6 (con 10.000 simulaciones de Monte Carlo en todos los casos, $N = 8$ para los mapas unimodales y $N = 4$ para el multimodal). Comparando dichas figuras se puede comprobar cómo cada estimador alcanza el CRLB esencialmente cuando su tasa de aciertos se aproxima al 100 % (es decir, cuando la secuencia simbólica se encuentra siempre bien estimada), como ya se había observado en el Capítulo 4.

Además, en general se aprecia que existe una relación directa entre la tasa de aciertos y el MSE: el algoritmo que proporciona una estima de la secuencia simbólica con un menor número de errores suele ser el que presenta el mejor MSE. Por ejemplo, las tasas de acierto de la Figura 5.7(c) son un fiel reflejo del MSE de los distintos estimadores, mostrado en la Figura 5.6(c): los estimadores HC-ML(6) y HC-ML(8) incrementan en gran medida la tasa de aciertos con respecto al FB-ML y el HC-ML(0), lo que implica una mejora notable del MSE. Una relación similar entre la tasa de aciertos y el MSE se aprecia comparando la Figura 5.7(a) con la Figura 5.6(a). En esta ocasión apenas existen diferencias entre las tasas de acierto de los distintos estimadores, lo que da lugar a diferencias igualmente pequeñas en su MSE.

No obstante esta relación no siempre se cumple, ya que puede haber errores en el itinerario más graves que otros, esto es, que tienen un mayor impacto sobre el MSE

final. Esto provoca que en ocasiones un estimador con una tasa de errores ligeramente superior presente un MSE inferior (compárense por ejemplo en las figuras 5.7(d) y 5.6(d) el FB-ML, con mayor tasa de aciertos, y el HC-ML(0), con menor MSE para SNR $<$ 25dB) o que dos estimadores con una tasa de errores similar tengan un MSE muy distinto, como ocurre con el HC-ML(8) y el FB-ML en las figuras 5.7(b) y 5.6(b): la diferencia en la tasa de aciertos dentro del rango de 10 a 35 dB de SNR es marginal, mientras que la ganancia en MSE del HC-ML(8) es importante.

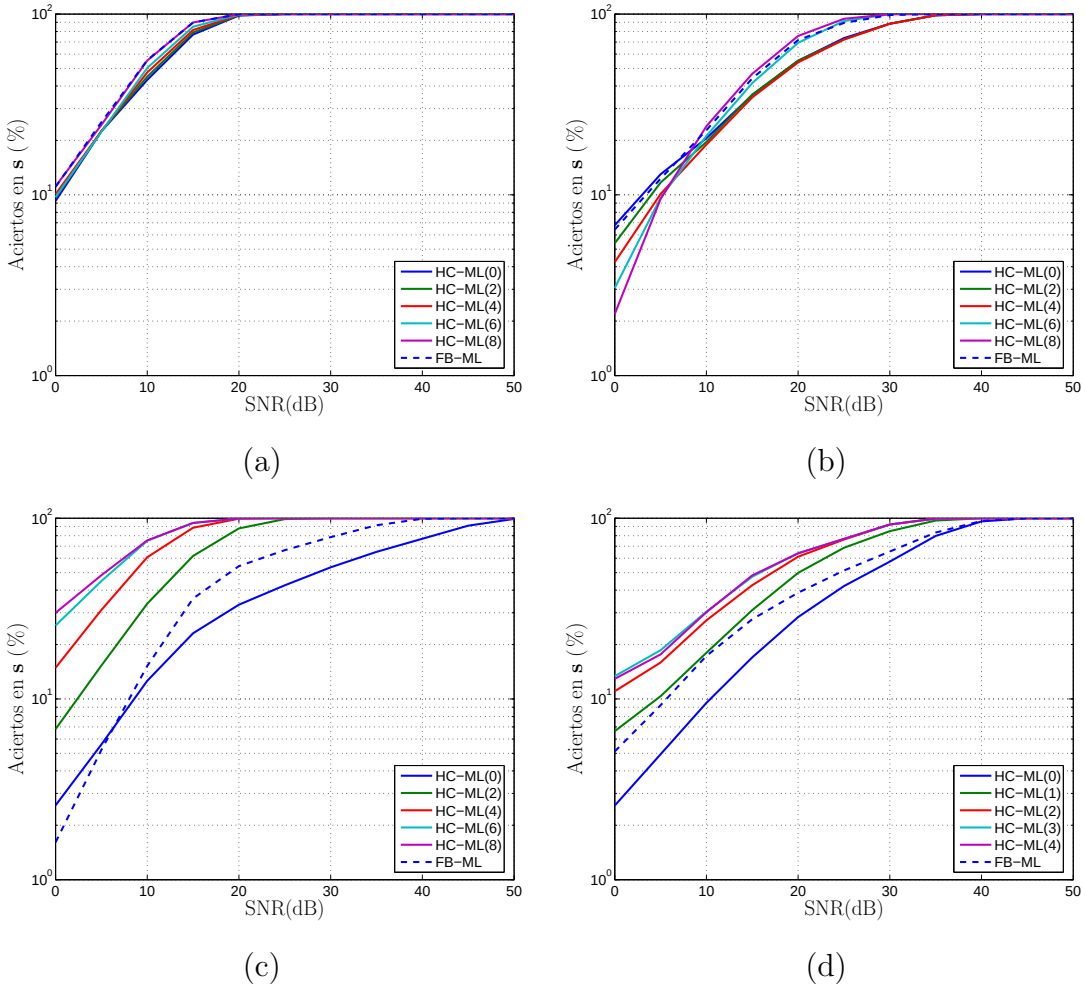
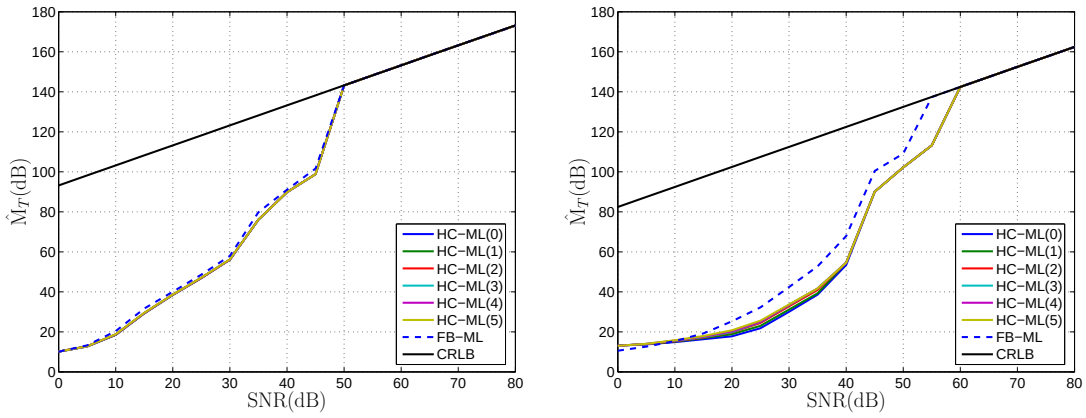


Figura 5.7: Tasa de aciertos en los símbolos del itinerario de los estimadores FB-ML y HC-ML(k) para los mismos ejemplos de la Figura 5.6. (a) S-TM con $\beta = 1{,}5$, $x[0] = 0{,}38316$ y $N = 8$. (b) SK-TM con $c = 0{,}1$, $x[0] = 0{,}2319$ y $N = 8$. (c) Mapa Markov2 de la Figura 2.5(a) con $x[0] = 0{,}99582$ y $N = 8$. (d) Mapa PWL1 de la Figura 2.5(b) con $x[0] = 0{,}89734$ y $N = 4$.

Por último, para concluir el estudio de los estimadores HC-ML y FB-ML, en la Figura 5.8 se muestran dos ejemplos del MSE para secuencias "largas" ($N = 29$) usando

de nuevo el SK-TM. Puede apreciarse cómo tanto el FB-ML como el HC-ML alcanzan el CRLB, aunque, al contrario de lo observado en general para secuencias cortas, el rendimiento del FB-ML es superior al del HC-ML(k). Esto es debido simplemente a que los valores de k utilizados ($0 \leq k \leq 5$) son mucho menores que N, por lo que el rendimiento de los distintos estimadores HC-ML es esencialmente idéntico al del HC-ML(0), pudiendo ser su MSE muy superior al del estimador ML.



(a) (b)

Figura 5.8: MSE total de la señal estimada para secuencias "largas" ($N = 29$) generadas usando el SK-TM. (a) $x[0] = 0{,}4322$ y $c = 0{,}5$. (b) $x[0] = 0{,}5926$ y $c = 0{,}9$.

Para finalizar, se ha analizado igualmente el comportamiento del estimador HC-MAP(k). Sin embargo, la conclusión obtenida es que, excepto para $k = N$ (esto es, para el estimador MAP), no existen diferencias apreciables con el rendimiento del estimador HC-ML(k). En consecuencia, no parece que resulte interesante su utilización, de modo que no se presentan gráficas con el valor de su MSE.

5.7.2. Estimadores más Sofisticados (SAGE y VA)

En esta sección se estudia el comportamiento de los estimadores basados en los algoritmos SAGE y de Viterbi. En primer lugar se analiza el rendimiento para secuencias cortas de los dos algoritmos SAGE propuestos: SAGE1, que estima $\mathbf{s}$ recursivamente y luego aplica el Algoritmo 3.3 para estimar $x[0]$ con el itinerario encontrado previamente; y SAGE2, que estima recursivamente tanto $\mathbf{s}$ como $x[0]$. En la Figura 5.9 se presentan dos ejemplos para el SK-TM con $N = 6$ y dos valores del parámetro de bifurcación, $c = 0{,}3$ y $c = 0{,}9$, en los que se compara el MSE de $\hat{x}[0]$ para los algoritmos SAGE1 y SAGE2 con el del HC-ML(0).

El comportamiento típico es el mostrado en la Figura 5.9(a): para SNRs muy bajas la calidad de los tres estimadores es esencialmente la misma; para SNRs intermedias existe una fuerte ganancia de los estimadores basados en el algoritmo SAGE frente al HC-ML, que se traduce en que habitualmente alcanzan el CRLB antes; y para SNRs

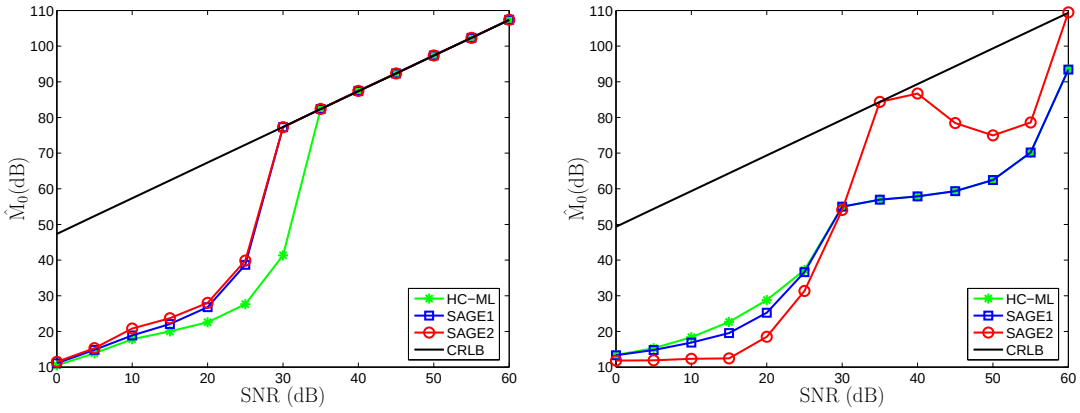


Figura 5.9: Comparación del MSE de los algoritmos SAGE1 y SAGE2 con el del HC-ML(0) para el SK-TM con $N = 6$. (a) $c = 0{,}3$ y $x[0] = 0{,}24859$. (b) $c = 0{,}9$ y $x[0] = 0{,}65631$.

altas todos presentan el mismo MSE, dado por el CRLB. Respecto al SAGE1 y SAGE2, su calidad es similar, aunque usualmente el rendimiento del SAGE2 es ligeramente superior al del SAGE1 desde el punto de vista del MSE.

Sin embargo, en ocasiones el comportamiento del HC-ML puede ser mejor que el de los algoritmos SAGE1 y SAGE2 (especialmente para una baja SNR) y el MSE de estos diferir notablemente, como se muestra en la Figura 5.9(b). En esta ocasión para SNR < 30 dB el HC-ML(0) presenta un valor de $\hat{M}_0$(dB) ligeramente superior al del SAGE1, y mucho mejor que el del SAGE2. No obstante, para SNR > 30 dB su comportamiento cambia radicalmente: el rencimiento del HC-ML(0) y SAGE1 es esencialmente idéntico, mientras que el SAGE2 mejora sustancialmente, siendo a partir de entonces claramente superior a los otros dos. Este comportamiento anómalo puede achacarse a que la estima HC-ML(0) inicial de $x[0]$ y $\mathbf{s}$ no es suficientemente buena para SNRs tan bajas y el SAGE2 converge a algún mínimo local.

Para corroborar el buen comportamiento de los algoritmos SAGE y su ganancia con respecto al HC-ML(0) en media, en la Tabla 5.7.2 se muestra el MSE promedio de $\hat{x}[0]$ para $N = 6$, $c = 0{,}9$ y 1.000 condiciones iniciales para cada valor de SNR. Como puede apreciarse, el SAGE2 es el que mejor funciona en general, proporcionando una ganancia de hasta 0,8 dB con respecto al SAGE1 para una SNR $= 15$ dB, y de hasta 4,9 dB con respecto al HC-ML(0) para una SNR $= 20$ dB.

A continuación se analiza el rendimiento del estimador basado en el algoritmo de Viterbi, también para secuencias cortas ($N = 4$). En la Figura 5.10 se muestran dos ejemplos del VA con $r = 1$ (esto es, con sólo dos estados en el "trellis"), comparándolo con el HC-ML(0) y el ML, para el SK-TM con $c = 0{,}2$ y el BSM. En ambos casos el rendimiento del VA es muy superior al del HC-ML(0), y similar al del ML con tan sólo una fracción de su coste computacional.

SNR (dB)	$\hat{M}_0$(dB)			
	HC-ML	SAGE-1	SAGE-2	ML
0	15.6	15.7	15.0	14.7
5	18.7	18.9	19.0	18.7
10	22.7	24.0	24.6	24.3
15	27.2	30.0	30.8	30.4
20	31.9	36.2	36.8	36.8
25	38.3	42.0	42.6	43.3
60	88.6	89.0	89.0	89.0

Tabla 5.5: Comparación del MSE de los estimadores HC-ML(0), SAGE1, SAGE2 y ML para el SK-TM con $c = 0{,}9$ y $N = 6$. Para cada valor de SNR se han seleccionado 1.000 valores de $x[0]$ de manera aleatoria.

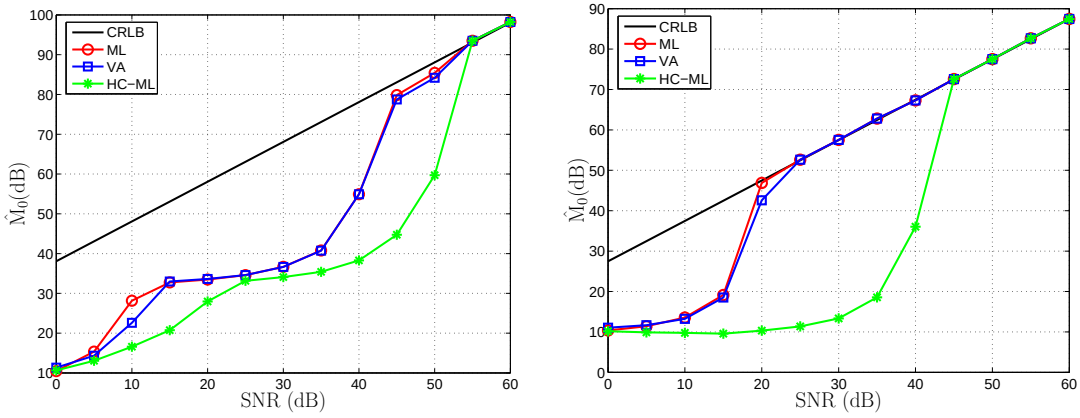


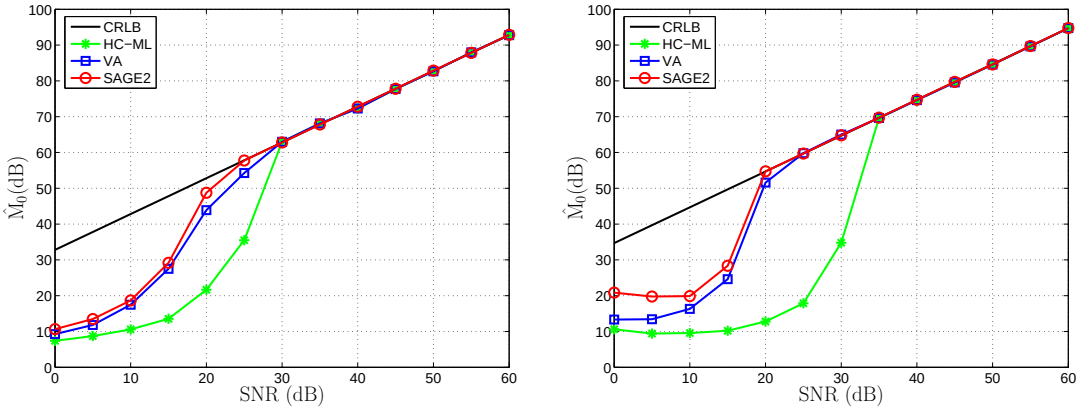
Figura 5.10: Comparación del MSE del algoritmo de Viterbi ($r = 1$) con el ML y el HC-ML(0). (a) SK-TM con $c = 0{,}2$, $x[0] = 0{,}1934$ y $N = 4$. (b) BSM con $x[0] = 0{,}4776$ y $N = 4$.

Para confirmar el buen rendimiento del estimador basado en el VA, en la Tabla 5.7.2 se muestra de nuevo su rendimiento en promedio (en esta ocasión para el BSM con $N = 4$) usando 1.000 condiciones iniciales distintas para cada valor de SNR al igual que con los algoritmos SAGE. En este caso se compara el MSE del VA con distintos valores de r: $r = 1$ ($R = 2$ estados en el "trellis"), $r = 2$ ($R = 4$ estados), y $r = 3$ ($R = 8$ estados). Como se puede apreciar, se consiguen hasta 0,4 dB de ganancia para $r = 3$ con respecto a $r = 2$ y hasta 1,5 dB con respecto a $r = 1$ para SNR = 10 dB. Sin embargo, la mayor ganancia se obtiene comparando cualquiera de los tres estimadores basados en el VA con el HC-ML(0): hasta 8,7 dB para SNR = 20 dB. Esto es debido a que en dicho punto el VA ya ha alcanzado el CRLB, y el HC-ML(0) aún no.

SNR (dB)	$\hat{M}_0$(dB)					
	HC-ML(0)	VA ($r=1$)	VA ($r=2$)	VA ($r=3$)	ML	CRLB
0	10.5	10.9	11.3	11.4	11.6	20.6
5	14.2	14.2	15.1	15.4	15.3	25.6
10	19.8	19.5	20.6	21.0	21.0	30.6
15	24.7	30.0	30.9	31.2	30.6	35.6
20	31.9	40.6	40.6	40.6	40.6	40.6
25	45.5	45.5	45.5	45.5	45.5	45.6
60	80.4	80.4	80.4	80.4	80.4	80.6

Tabla 5.6: Comparación del MSE de los estimadores HC-ML(0), VA ($r = 1,\ 2,$ y 3) y ML para el BSM con $N = 4$. Para cada valor de SNR se han seleccionado 1.000 valores de $x[0]$ de manera aleatoria.

Para finalizar esta sección, a continuación se compara el rendimiento e los estimadores basados en los algoritmos SAGE y VA, tanto para secuencias cortas ($N = 6$), Figura 5.11, como "largas" ($N = 29$), Figura 5.12, usando el SK-TM con diferentes valores del parámetro de bifurcación y condiciones iniciales. Como puede verse en ambas figuras, el rendimiento del SAGE2 parece ser ligeramente superior al del VA con $r = 1$, aunque su coste computacional también es ligeramente superior, ya que en general son necesarias varias iteraciones sobre la secuencia simbólica (por ejemplo, en la Figura 5.11 se han llevado a cabo 10 iteraciones), y además el SAGE2 requiere estimar $x[0]$ tras cada cambio en la secuencia simbólica, mientras que el VA únicamente lleva a cabo dicha estimación en el paso final.



(a) (b)

Figura 5.11: Comparación del MSE de los algoritmos SAGE2, de Viterbi ($r = 1$) y HC-ML(0) para el SK-TM con secuencias cortas ($N = 6$). (a) $c = 0{,}2$ y $x[0] = 0{,}09197$. (b) $c = 0{,}9$ y $x[0] = 0{,}96091$.

Por último, en relación con la probabilidad de error en los símbolos (no mostrada) las conclusiones son las mismas que las de la Sección 5.7.1: todos los estimadores alcanzan el CRLB cuando su tasa de acierto en los símbolos tiende al 100 %, y en general aquel estimador que tiene una mejor tasa de aciertos proporciona un menor MSE.

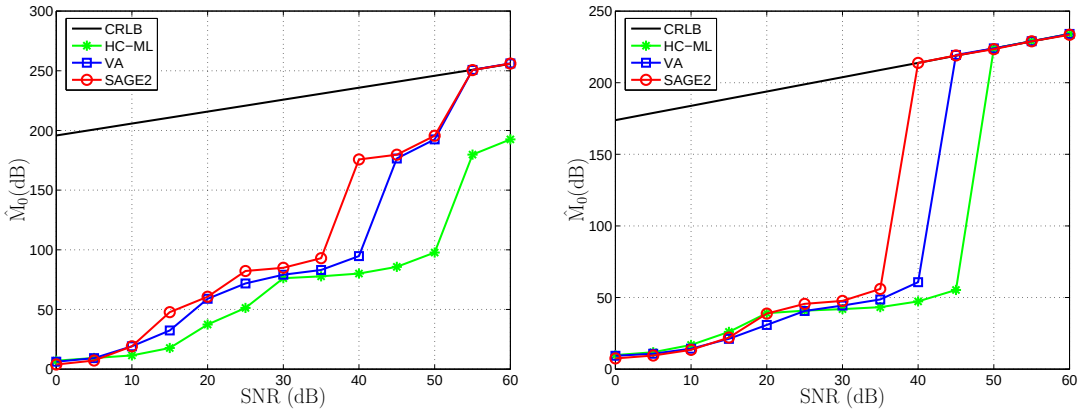


Figura 5.12: Comparación del MSE de los algoritmos SAGE2, de Viterbi ($r = 1$) y HC-ML(0) para el SK-TM con secuencias "largas" ($N = 29$). (a) $c = 0{,}2$ y $x[0] = 0{,}98525$. (b) $c = 0{,}75$ y $x[0] = 0{,}32004$.

5.8. Discusión

En este capítulo se ha analizado el coste computacional de los estimadores óptimos presentados en los capítulos anteriores, y se han propuesto diversos algoritmos computacionalmente eficientes. En primer lugar, se han planteado algoritmos sencillos y con un coste muy reducido, como el HC-ML(k) o la extensión del algoritmo recursivo de Papadopoulos y Wornell, que se ha denominado algoritmo FB-ML. A continuación, se han propuesto otras dos clases de estimadores basados en técnicas ampliamente conocidas y utilizadas en problemas de comunicaciones digitales: los algoritmos E-M y SAGE, y el algoritmo de Viterbi. Por último, se ha comparado el rendimiento de estos estimadores entre sí, así como con los estimadores óptimos de los capítulos 3 y 4 siempre que ha sido posible (esto es, para secuencias cortas). Las medidas de calidad utilizadas han sido nuevamente el MSE y la probabilidad de error en los símbolos del itinerario, y las principales conclusiones obtenidas han sido las siguientes:

1. Se ha demostrado que el número de itinerarios válidos crece exponencialmente con la longitud de la secuencia para cualquier mapa caótico unidimensional. Además, para garantizar la obtención de los estimadores óptimos se deben explorar necesariamente todos los itinerarios, no pudiéndose descartar ninguno a priori. En consecuencia, resulta inevitable que el coste computacional de los estimadores óptimos dependa exponencialmente de la longitud de la señal a estimar.

2. Cuando se dispone de una estima del itinerario, para obtener los estimadores ML o MAP solamente es necesario calcular un único estimador local, cuyo coste es cuadrático con la longitud de la señal. Por consiguiente, los esfuerzos a la hora de diseñar algoritmos eficientes deben concentrarse en conseguir buenas estimas de la secuencia simbólica de un modo computacionalmente eficiente.

3. El estimador MS requiere el promediado de las estimas locales correspondientes a todos los itinerarios válidos, o al menos de todos aquellos para los que $p(x[0]|\mathbf{y}, \mathbf{s}_i)$ toma valores significativos. El problema de encontrar este conjunto de itinerarios resulta mucho más complicado y costoso que la obtención de una única secuencia simbólica adecuada. Por lo tanto, no se contempla en este capítulo y se mantiene como una posible línea futura.

4. El algoritmo subóptimo más sencillo que se puede proponer es el HC-ML(0), consistente en usar el itinerario obtenido directamente aplicando un umbral duro a las observaciones como estima de la secuencia simbólica. Aunque este algoritmo proporciona resultados cada vez más cercanos a los del estimador ML exacto conforme aumenta la SNR, alcanzando el CRLB asintóticamente, su rendimiento para SNRs medias/bajas resulta claramente mejorable.

5. El algoritmo HC-ML(k) mejora el rendimiento del HC-ML(0) para secuencias cortas considerando, junto con el itinerario "ruidoso", todos aquellos itinerarios resultantes de modificar alguno de los k símbolos menos fiables. En este sentido, el parámetro k permite introducir un compromiso entre rendimiento y coste computacional. Para secuencias "largas" no merece la pena usar valores de $k \neq 0$, puesto que k debe ser generalmente mucho menor que N, y no se consigue ninguna mejora en el MSE.

6. El algoritmo FB-ML, desarrollado originalmente por Papadopoulos y Wornell en [Papado1993, Papado1995] como estimador ML exacto para el TM, se puede extender de manera sencilla y efectiva a cualquier otro mapa caótico unidimensional como algoritmo subóptimo, obteniéndose resultados ligeramente peores a los del HC-ML(k) en general para secuencias cortas, pero ligeramente mejores para secuencias "largas".

7. El algoritmo E-M, empleado con frecuencia para implementar el estimador ML en problemas complicados y/o con una elevada carga computacional, no resulta útil en esta ocasión, ya que no permite evitar el crecimiento exponencial del coste computacional del estimador ML con la longitud de la señal caótica. Sin embargo, se ha mostrado cómo una variante del mismo, la familia de algoritmos SAGE, que admiten una implementación sin datos ocultos, consigue una importante reducción del coste computacional a la par que un rendimiento cercano al óptimo.

8. La utilización de la versión exacta del algoritmo de Viterbi tampoco permite reducir el coste computacional, ya que los sistemas caóticos pueden verse como filtros

IIR no lineales, de modo que el "trellis" de decodificación requeriría un número de estados igual al de itinerarios válidos. No obstante, se ha comprobado que la aplicación del VA con un número de estados extremadamente reducido (incluso con sólo dos estados) proporciona muy buen resultado debido a la convergencia de las trayectorias de cualquier mapa caótico en la iteración hacia atrás.

9. Los dos métodos que proporcionan mejores resultados son los basados en los algoritmos SAGE y de Viterbi. En ambos casos el MSE obtenido para secuencias cortas resulta similar al del estimador ML exacto para cualquier valor de SNR, con un coste computacional ligeramente superior al de los algoritmos HC-ML(k) o FB-ML, pero notablemente inferior al de la "búsqueda exhaustiva" utilizada en el Capítulo 3. Para secuencias "largas" el estimador ML exacto resulta imposible de calcular, pero los estimadores basados en el SAGE o en el VA mejoran sustancialmente el rendimiento de algoritmos más sencillos como el HC-ML.

10. Aunque los algoritmos anteriores han sido diseñados originalmente para encontrar el estimador ML, todos ellos (con la excepción del FB-ML) se pueden aplicar a la obtención del estimador MAP mediante una sencilla modificación consistente en incluir un término en sus respectivas funciones de coste que tenga en cuenta la FDP a priori de la secuencia simbólica.

11. A pesar de realizar los desarrollos a lo largo del capítulo exclusivamente para mapas PWL, todos los algoritmos propuestos para obtener una estima del itinerario son directamente aplicables a cualquier mapa unidimensional sea o no PWL. La única dificultad para obtener la estima ML o MAP de señales generadas por mapas no PWL reside en estimar $x[0]$ una vez que se dispone de la secuencia simbólica, para lo cual se puede recurrir a las técnicas de rejilla o los métodos iterativos locales mencionados en los capítulos 3 y 4.

Para finalizar, respecto al uso de los estimadores computacionalmente eficientes propuestos con mapas d-dimensionales, el principal problema es de nuevo el coste computacional. Los algoritmos HC-ML(0) y FB-ML son directamente extensibles a dicha clase de mapas sin un aumento apreciable del mismo. Sin embargo, el resto de algoritmos van a experimentar un incremento exponencial en su coste con la dimensión del mapa caótico, d. Por ejemplo, para un mapa PWL con M intervalos por dimensión, el coste computacional del algoritmo HC-ML(k) sería $\Theta(M^{kd}N^2)$ en el peor caso, el de los algoritmos SAGE1 y SAGE2 sería $\Theta(n_v M^d N^3)$, y el estimador basado en el VA debería utilizar un "trellis" con M^{rd} estados.