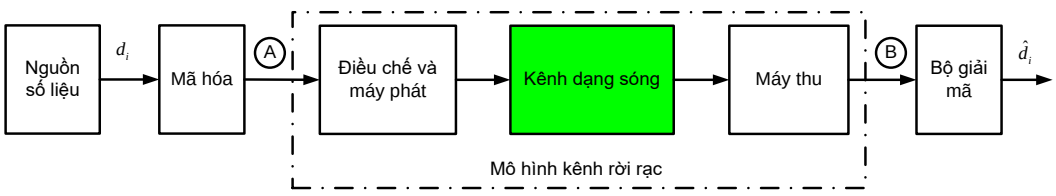
Chương 15
MÔ HÌNH KÊNH RỜI RẠC

15.1. Mở đầu

Trong chương này, ta xét các giải pháp khác cho bài toán lập mô hình kênh nhằm có được mô hình mô phỏng hiệu quả. Trong phần trước, kênh đã được định nghĩa ở dạng tạp âm, nhiễu và các nhiễu loạn khác, chúng kết hợp với tín hiệu phát tạo ra dạng sóng tạp âm và méo tại đầu vào máy thu. Tín hiệu phát, tập âm, nhiễu và các nhiễu loạn kênh khác, tất cả đều được biểu diễn bởi các mẫu của các dạng sóng. Kết quả là mô phỏng mức dạng sóng, số liệu được xử lý trên cơ sở *từng mẫu*. Tại đây, ta xét việc phân chia hóa hệ thống theo hướng loại bỏ nhiều phần được dùng trong mô phỏng mức dạng sóng. Kết quả là ta được mô hình kênh rời rạc trong đó việc mô phỏng trên cơ sở *từng ký hiệu*. *Động cơ* để thay thế mô hình kênh dạng sóng bằng mô hình kênh rời rạc là *tốc độ mô phỏng*. Ta sẽ thấy rằng, mô hình kênh rời rạc là sự trừu tượng hóa của kênh vật lý (dạng sóng), trong đó kênh được đặc tính hóa đầy đủ bằng *một tập nhỏ* các tham số. Xác định rõ các tham số này góp phần quan trọng vào quá trình mô hình hóa và phải được thực hiện thông qua đo đạc trên kênh vật lý, hoặc thông qua việc dùng một mô hình mức dạng sóng. Một khi có được mô hình khả tin, thì mô hình kênh dạng sóng có thể được thay bằng mô hình kênh rời rạc tương đương.

Mô hình cơ bản của một hệ thống truyền thông được minh họa ở hình 15.1 gồm có: nguồn số liệu rời rạc, bộ mã hóa kênh kiểm soát lỗi, bộ điều chế, máy phát, kênh vô tuyến, máy thu, bộ giải mã hóa kênh. Tùy vào ứng dụng và tính chi tiết của mô phỏng, các phần tử khác như các bộ cân bằng, bộ đan xen, bộ giải đan xen, bộ đồng bộ sóng mang và đồng bộ ký hiệu có thể có trong mô hình hệ thống. Bộ điều chế sắp xếp các ký hiệu (hoặc chuỗi ký hiệu) đầu vào thành một dạng sóng đầu ra. Dạng sóng phải chịu nhiều ảnh hưởng trong kênh. Các ảnh hưởng phải tính đến là tạp âm, giới hạn băng tần, nhiễu và pha đỉnh, chúng đều được đặc trưng hóa bằng các dạng sóng. Đầu vào máy thu cũng là một dạng sóng, là dạng sóng phát kết hợp với các ảnh hưởng xấu của kênh. Vai trò của máy thu là quan sát sóng đầu vào trong khoảng thời gian một ký hiệu hoặc một chuỗi các ký hiệu, và xác định chuỗi ký hiệu phát. Kênh trong trường hợp này được coi là kênh dạng sóng, vì đầu vào/ra của kênh đều được đặc tính là các dạng sóng. Đặc biệt lưu ý rằng, tất cả các thành phần hệ thống đều được đặc tính bởi các ánh xạ (sắp xếp) tất định, ngoại trừ kênh. Kênh thực hiện sắp xếp ngẫu nhiên (hay stochastic) tín hiệu vào thành tín hiệu ra của kênh.



Hình 15.1: Hệ thống truyền thông với mô hình kênh rời rạc

Thuật ngữ "*mô hình kênh rời rạc*" DCM được dùng để bao hàm tất cả các phần tử của hệ thống truyền thông nằm giữa hai điểm A và B trong đó đầu vào tại A là một véc tơ các ký hiệu rời rạc (chuỗi đầu vào) $\mathbf{X} = [x_1, x_2, ..., x_k, ...]$ và đầu ra tại B là một véc tơ các ký hiệu rời rạc khác (chuỗi đầu ra) $\mathbf{Y} = [y_1, y_2, ..., y_k, ...]$.

Thường điểm A là đầu ra bộ mã hóa kênh (đầu vào bộ điều chế), điểm B là đầu vào bộ giải mã như được thấy ở hình 15.1. Lưu ý rằng, bằng cách phân chia hóa như hình 15.1, thì bộ điều chế, máy phát, kênh dạng sóng, máy thu đều là các bộ phận của mô hình kênh rời rạc. Quan hệ giữa các véc tơ vào/ra (X/Y) của kênh rời rạc đều bị ảnh hưởng bởi các thành phần hệ thống (như bộ lọc) và các nhiễu loạn ngẫu nhiên do kênh vật lý (kênh dạng sóng). Trong hệ thống truyền thông nhị phân dùng máy thu quyết định cứng, thì các phần tử của X và Y đều là các chuỗi nhị phân. Trong trường hợp máy thu quyết định mềm, các phần tử của Y sẽ từ bảng mẫu tự M-ary. Các lỗi truyền dẫn do tính không hoàn hảo của các phần tử hệ thống giữa điểm A và điểm B bao gồm kênh vật lý, chúng gây ra Y khác X.

Các mô hình kênh rời rạc mô tả cơ chế tạo lỗi có thể xảy ra. Các mô hình này được chia làm hai loại: *(i)* Lớp những mô hình kênh rời rạc "*không nhớ*", được dùng để mô hình hóa các lỗi truyền dẫn hoặc mô hình hóa các chuyển tiếp từ đầu vào kênh tới đầu ra kênh với giả định không có tương quan thời gian trong cơ chế chuyển tiếp (nghĩa là các xác suất chuyển tiếp vào/ra cho ký hiệu đầu vào kênh thứ *n* không bị ảnh hưởng bởi những gì xảy ra cho bất kỳ các ký hiệu đầu vào khác). Các mô hình này thường ứng dụng cho các kênh không có ISI hoặc pha đinh và tạp âm là AWGN. Trong hệ thống nhị phân quyết định cứng, giả định này có nghĩa là các lỗi bit không tương quan nhau. *(ii)* Lớp các mô hình kênh rời rạc "*có nhớ*", là lớp các mô hình đáng quan tâm hơn để ứng dụng vào tình huống ở đó việc chuyển tiếp các ký hiệu vào/ra bị *tương quan theo thời gian*. Trong trường hợp này, xác suất lỗi đối với ký hiệu thứ *n* phụ thuộc vào có xảy ra lỗi ở lần truyền dẫn các ký hiệu trước đó hay không. Kênh pha đinh thường gặp phải trong truyền thông vô tuyến là minh họa rõ nét cho kênh có các *lỗi tương quan*. Các lỗi sẽ có xu hướng xuất hiện ở dạng các cụm khi kênh vô tuyến là kênh *pha đinh sâu* và những kênh này được coi là các kênh lỗi *cụm* hay kênh có *nhớ.*

Các mô hình kênh rời rạc là những mô hình xác suất có hiệu quả tính toán cao hơn so với các mô hình kênh dạng sóng. Hiệu quả được gia tăng nhờ hai nhân tố (tốc độ lấy mẫu và mức độ trừu tượng): *(i)* Tốc độ lấy mẫu, các mô hình kênh rời rạc được mô phỏng tại tốc độ ký hiệu trong khi đó các mô hình mức dạng sóng thường được mô phỏng tại *tốc độ* từ 8 đến 16 lần tốc độ ký hiệu. Điều này làm giảm đáng kể gánh nặng tính toán. *(ii)* Mức độ trừu tượng, trong khi mỗi khối riêng rẽ được mô phỏng chi tiết trong mô hình mức dạng sóng thì mô hình kênh rời rạc được *trừu tượng* ở mức cao hơn. Mức trừu tượng này sẽ làm giảm tải tính toán. Hai nhân tố này góp phần giảm thời gian mô phỏng.

Trong các trường hợp đơn giản, có thể rút ra được các mô hình kênh rời rạc theo phép giải tích từ các mô hình của các thành phần cơ bản giữa đầu vào A và đầu ra B của kênh rời rạc. Tuy nhiên, hầu hết các trường hợp, các mô hình kênh rời rạc được rút ra từ các mẫu lỗi đo hoặc mô phỏng giữa điểm A và điểm B. Các mô hình kênh rời rạc được dùng để thiết kế và

phân tích ảnh hưởng của các thành phần *bên ngoài* phần hệ thống giữa điểm A và B. Ví dụ, các bộ mã hóa kiểm soát lỗi, bộ lập mã nguồn và bộ đan xen.

Mô mô hình hóa các kênh rời rạc *không nhớ* là quá trình tương đối dễ dàng. Ví dụ, trường hợp kênh đối xứng rời rạc không nhớ có đầu vào/ra nhị phân, tất cả cần phải đặc tính hóa kênh là một con số, cụ thể là xác suất lỗi bit. Việc mô phỏng kênh bao gồm rút ra một con số ngẫu nhiên và so sánh với ngưỡng để quyết định bit truyền qua kênh có bị lỗi hay không. Theo đó, để mô phỏng quá trình truyền dẫn hàng triệu bit qua kênh, ta phải tạo ra hàng triệu con số ngẫu nhiên phân bố đều độc lập và so sánh với ngưỡng. Vì vậy, toàn bộ hệ thống được biểu diễn bởi kênh này được mô phỏng một cách hiệu quả. (So sánh mô phỏng này với mô phỏng mức dạng sóng - bao gồm việc tạo ra nhiều triệu mẫu để biểu diễn các dạng sóng và xử lý các mẫu này thông qua tất cả các khối chức năng của hệ thống).

Mô hình hóa các kênh rời rạc *có nhớ* là khó khăn hơn nhiều. Cơ chế tạo lỗi tương quan theo thời gian thường được mô hình hóa bằng một chuỗi Markov thời gian rời rạc, trong đó *mô hình trạng thái được dùng để đặc tính hóa các trạng thái khác nhau của kênh* và một *tập các xác suất truyền được dùng để bắt giữ tiến triển các trạng thái kênh*. Mỗi trạng thái được liên kết với một tập các xác suất truyền ký hiệu từ đầu vào tới đầu ra. Vì vậy, mô hình này phức tạp hơn và cần có nhiều tham số hơn so với kênh không tương quan. Cấu trúc mô hình và các giá trị của các tham số được ước tính từ các mẫu lỗi đo hoặc mô phỏng. Việc mô phỏng kênh rời rạc bao gồm tạo số ngẫu nhiên trước khi truyền dẫn mỗi ký hiệu nhằm xác định trạng thái kênh và sau đó rút ra một số ngẫu nhiên khác để xác định truyền từ đầu vào tới đầu ra. Trong khi thủ tục lập mô hình và ước tính tham số là phức tạp, thì việc mô phỏng cho mô hình Markov lại rất hiệu quả. Ta sẽ đề cập thêm những ý tưởng này trong các phần tiếp theo.

Theo đó, tiêu điểm của chương là triển khai các mô hình kênh rời rạc. Các mô hình kênh rời rạc là hấp dẫn vì dùng chúng giảm được rất nhiều độ phức tạp tính toán khi thực hiện mô phỏng. *Các tham số của mô hình có thể được xác định từ số liệu đo đạc hoặc từ kết quả mô phỏng mức dạng sóng*. Các mô hình kênh rời rạc được dùng rộng rãi trong các hệ thống truyền thông vô tuyến vì chúng được dùng để lập mô hình một cách hiệu quả các kênh pha đinh. Như trên đã đề cập, các mô hình kênh rời rạc là một trừu tượng hóa của các mô hình dạng sóng trong đó chúng đặc trưng hóa các đặc tính đầu vào/ra của kênh nhưng không lập mô hình chức năng vật lý của kênh. Thông qua tính trừu tượng này, gánh nặng tính toán được giảm.

Trước hết, ta xét cho mô hình hai trạng thái để thiết lập khái niệm ma trận chuyển dịch trạng thái, véc tơ phân bố trạng thái và ma trận tạo lỗi. Một vài kỹ thuật khác để xác định ma trận phân bố trạng thái bền được đề cập. Sau đó, các khái niệm này được mở rộng cho mô hình N trạng thái bao gồm việc triển khai các kỹ thuật để mô phỏng kênh một cách hiệu quả trên cơ sở mô hình kênh rời rạc.

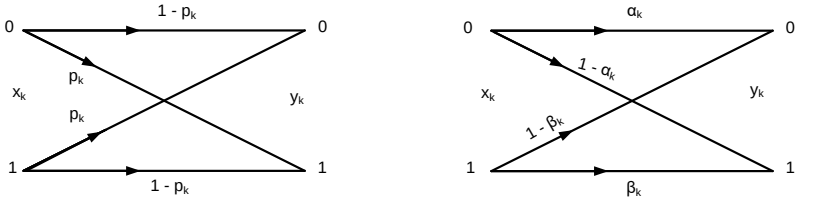
Tiếp đến, ta sẽ xét cho mô hình Gilbert 2 trạng thái và mô hình Fritchman N trạng thái như là các ví dụ của mô hình Markov. Với mục đích tính toán thì Mô hình Fritchman là hấp dẫn vì ma trận chuyển dịch trạng thái chứa một số lượng lớn số 0 (tương đối thưa). Mô hình Fritchman cũng cho thấy hiệu quả trong việc mô hình hóa các môi trường kênh pha đinh.

Trọng tâm của chương là xử lý triển khai các mô hình kênh rời rạc trên cơ sở dữ liệu đo đạc hoặc dữ liệu từ mô phỏng mức dạng sóng. Công cụ cơ bản được dùng để ước tính tham số là thuật toán Baum-Welch. Việc triển khai thuật toán Baum-Welch được phác thảo ngắn gọn và việc triển khai mô phỏng Matlab cho thuật toán Baum-Welch cũng được trình bày. Đề cập việc lấy tỉ lệ ngăn chặn mất giá trị, là một mô hình Markov tương đương khối, mô hình này có thể được ước tính hiệu quả hơn mô hình Markov tổng quát.

Kết thúc chương bằng ba một ví dụ và hai trường hợp điển hình tóm tắt các kỹ thuật để ước tính các mô hình Markov và các mô hình Markov tương đương khối. Ví dụ 15.3 tập trung vào việc tạo một véc tơ lỗi được cho mô hình Markov. Tính hợp lý của kết quả dựa trên xác suất chiếm giữ trạng thái và xác suất lỗi. Trường hợp thứ nhất dùng thuật toán Baum-Welch để ước tính mô hình kênh. Tính hợp lệ gồm việc so sánh con số $\Pr\{0^m|I\}$ cho cả véc tơ lỗi gốc và véc tơ lỗi được tao ra bởi mô hình được ước tính. Trường hợp thứ 2 tương tự với trường hợp thứ nhất ngoại trừ mô hình Markov khối tương được dùng. Ví dụ 15.3 và hai trường hợp điển hình này có tác dụng như là một hướng dẫn sử dụng các công cụ được triển khai trong chương này.

15.2. Mô hình kênh rời rạc không nhớ

Trong kênh rời rạc *không nhớ*, việc sắp xếp từ đầu vào tới đầu ra là *tức thì* và được mô tả bằng một tập các xác suất truyền. Mô hình kênh rời rạc không nhớ đơn giản nhất là mô hình kênh nhị phân đối xứng (BSC), được minh họa ở hình 15.2(a). Đầu vào của kênh rời rạc là một chuỗi ký hiệu nhị phân, được ký hiệu là véc tơ X. Thành phần thứ k của véc tơ này là x_k, tương ứng với đầu vào kênh thứ k, và p_k tương ứng với xác suất lỗi truyền dẫn ký hiệu thứ k. Đối với kênh không nhớ, p_k không phụ thuộc vào k, vì vậy lỗi trên tất cả các ký hiệu đều bị ảnh hưởng bởi kênh với cùng một kiểu. (ở dạng tổng quát, nếu kênh có nhớ thì p_k là một hàm của p_{k-1}). Chuỗi M ký hiệu được xử lý thông qua kênh không nhớ bằng cách gọi mô hình kênh M lần liên tiếp. Với ký hiệu thứ k, đầu vào nhị phân là "0" thu đúng là "0" với xác suất là 1- p_k, và thu sai là "1" với xác suất lỗi là p_k. Đầu ra kênh thứ k được ký hiệu là y_k và chuỗi M ký hiệu của các đầu ra được ký hiệu là véc tơ Y.



Hình 15.2: Các mô hình kênh nhị phân

Mô hình kênh nhị phân đối xứng

Kênh là *đối xứng* trong đó các số 0 và số 1 đều bị ảnh hưởng bởi kênh theo cùng một kiểu. Lưu ý rằng kết quả của tính đối xứng là xác suất lỗi không phụ thuộc vào ký hiệu phát, vì vậy nguồn lỗi có thể được mô phỏng riêng biệt với nguồn thông tin (số liệu).

Đối với kênh nhị phân, quan hệ đầu vào/ra có thể được biểu diễn như sau:

$$\mathbf{Y} = \mathbf{X} \oplus \mathbf{E} \tag{15.1}$$

Trong đó X và Y là các véc tơ dữ liệu vào/ra, $\oplus$ là phép toán XOR, E là véc tơ lỗi. Cụ thể, E = $\{e_1, e_2, e_3,...\}$ là véc tơ nhị phân hay chuỗi có phần tử {0,1} trong đó $e_k = 0$ biểu thị phần tử thứ k của X, x_k được thu đúng ($y_k = x_k$), và $e_k = 1$ biểu thị phần tử thứ k của X, x_k được thu sai ($y_k \neq x_k$). Tham số xác định hiệu năng của mô hình kênh đối xứng nhị phân là xác suất lỗi P_E, có thể dễ dàng ước tính từ đo đạc hoặc từ việc phỏng hệ thống ở mức dạng sóng. Lưu ý rằng, đối với hệ thống nhị phân, ước tính xác suất lỗi theo Monte Carlo là trọng số *Hamming* của véc tơ lỗi E được chia cho N (số lượng các phần tử trong véc tơ lỗi E).

Lưu ý rằng, một khi biết được p_k, thì có thể tạo chuỗi lỗi tương ứng với N ký hiệu phát bằng cách gọi N lần bộ tạo số ngẫu nhiên phân bố đều và so sánh với số ngẫu nhiên với ngưỡng p. Cụ thể là:

$$e_k = \begin{cases} 1, & U_k \leq p_k \\ 0, & U_k > p_k \end{cases} \tag{15.2}$$

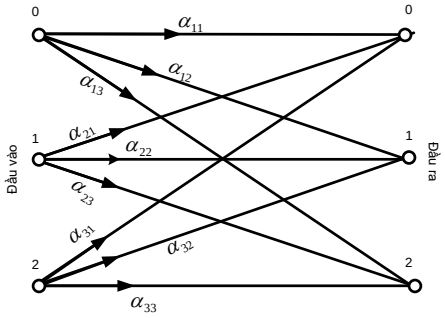
Trong đó U_k ký hiệu cho con số nhận được từ lần gọi thứ k bộ tạo số ngẫu nhiên phân bố đều. Nó quan trọng để hiểu rằng mô hình kênh rời rạc sẽ không tái tạo mẫu lỗi ban đầu, được dùng để xác định xác suất lỗi ban đầu p. Tuy nhiên, nó sẽ tái tạo một mẫu lỗi tương đương thống kê. Tính đơn giản của (15.2) giải thích lý do tại sao việc thay thế mô hình kênh rời rạc DCM cho phần hệ thống nằm giữa điểm A và B trong hình 15.1 dẫn đến hiệu quả tính toán.

Mô hình kênh nhị phân không đối xứng (tổng quát)

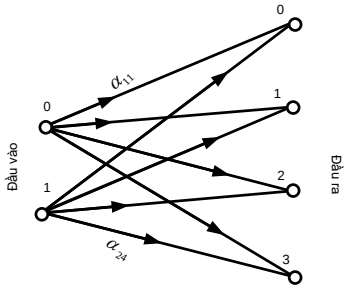
Mô hình phức tạp hơn một chút là *mô hình kênh nhị phân không đối xứng* trong đó xác suất lỗi có thể khác nhau đối với các số 0 và 1, như trong một số hệ thống thông tin quang. Điều này được minh họa ở hình 15.2(b). Mô hình này được đặc tính hóa bởi một tập bốn xác suất truyền từ đầu vào tới đầu ra, nhưng chỉ có hai trong số bốn xác suất truyền này α_k và β_k là độc lập nhau. Kênh là không đối xứng nếu $\alpha_k \neq \beta_k$.

Việc mở rộng mô hình này cho trường hợp đầu vào/ra thuộc về một M-ary ký tự của bảng mẫu tự là dễ dàng và được cho ở hình 15.3 với M = 3. Kênh được đặc tính hóa bằng một tập các xác suất truyền a_{ij}, $i, j = 0, 1, 2$ thể hiện xác suất có điều kiện:

$$\alpha_{ij} = \Pr\{\text{đầu ra} = j \,|\, \text{đầu vào} = i\} \tag{15.3}$$



Hình 15.3: Mô hình kênh rời rạc (3 đầu vào và 3 đầu ra)



Hình 15.4: Mô hình kênh rời rạc (hai đầu vào và 4 đầu ra)

Nếu tập các xác suất a_{ij} là hằng số, và vì vậy không phụ thuộc vào k, đầu ra kênh thứ k chỉ phụ thuộc vào đầu vào kênh thứ k. Với trường hợp này, mô hình ở hình 15.3 là không nhớ. Các xác suất truyền được ước tính bằng việc mô phỏng mức dạng sóng chi tiết trong đó đếm các truyền từ đầu vào thứ i tới đầu ra thứ j. Trong các mô hình này, các xác suất truyền được ước tính từ số liệu mô phỏng hoặc đo đạc là:

$$\alpha_{ij} = \frac{\textit{Số lần ký hiệu đầu vào thứ } \mathbf{i} \textit{ truyền đến đầu ra thứ } \mathbf{j}}{\textit{Số lần xuất hiện ký hiệu đầu vào thứ } \mathbf{i}} = \frac{n_{ij}}{N_i} \tag{15.4}$$

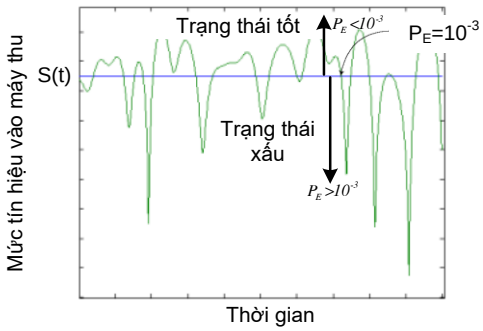
trong đó N_i là số lần xuất hiện ký hiệu đầu vào thứ i, n_{ij} là số lần ký hiệu đầu vào thứ i truyền tới đầu ra thứ j. Một ví dụ khác được cho ở hình 15.4, được dùng cho các kênh không nhớ trong đó đầu ra của kênh được lượng tử hóa thành nhiều mức từ các mức đầu vào, như trong giải mã quyết định mềm.

15.3. Mô hình Markov cho kênh rời rạc có nhớ

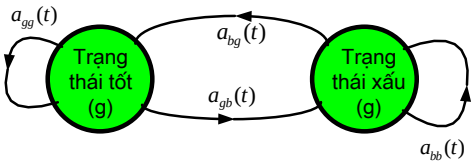
Với các kênh có nhớ, mô hình được dùng phổ biến nhất là mô hình Markov hữu hạn trạng thái rời rạc. Một vài lý do cho tính phổ biến của mô hình Markov. Các mô hình thuộc loại xử lý được theo phép giải tích (liên hệ với chương 1), lý thuyết của chúng dễ dàng được thiết lập theo lý thuyết thống kê, và chúng đã được ứng dụng thành công cho nhiều bài toán truyền thông quan trọng. Mô hình Markov đã được dùng để mô hình hóa đầu ra nguồn tin rời rạc như văn bản Anh ngữ (sự xuất hiện của chữ cái trong bảng mẫu tự Anh ngữ có thể được mô hình hóa như một chuỗi Markov). Tương tự, các giá trị mẫu của dạng sóng âm thanh hoặc hình ảnh cũng có thể được mô hình hóa bằng mô hình Markov (MM). Có thể trực tiếp ứng dụng các kỹ thuật mô hình hóa Markov để mô hình hóa và phân tích các kênh truyền thông rời rạc. Ngoài ra, các mô hình Markov cũng có thể được dùng để ước lượng dung lượng kênh rời rạc và thiết kế các kỹ thuật mã hóa kiểm soát lỗi tối ưu. Các mô hình Markov đã được dùng thành công để *đặc tính hóa* kênh pha đinh trong các hệ thống thông tin vô tuyến. Quan trọng nhất, các kỹ thuật tính toán hiệu quả là khả dụng để ước tính các tham số của các chuỗi Markov từ các mẫu lỗi đo đạc hoặc mô phỏng. Khi giới thiệu về hình Markov, trước hết ta xét mô hình hai trạng thái đơn giản. Sau đó, tổng quát hóa các kết quả cho mô hình N trạng thái.

15.3.1. Mô hình hai trạng thái

Để thiết lập tầng cho các mô hình kênh Markov, xét một kênh pha đinh trong đó cường độ tín hiệu thu trên mức ngưỡng hiệu năng chấp nhận được (khoảng thời gian khả dụng), và dưới mức ngưỡng (khoảng thời gian pha đinh sâu). Nếu ta chỉ quan tâm các điều kiện "ngưỡng trên" hoặc "ngưỡng dưới", thì ta có thể lập mô hình kênh vào một trong hai trạng thái như được thấy ở hình 15.5. Trong hình 15.5, ta có một trạng thái tốt *g* trong đó hiệu năng hệ thống chấp nhận được ($P_E < 10^{-3}$), và một trạng thái xấu *b* trong đó mức tín hiệu thu thấp đến mức làm cho xác suất lỗi không chấp nhận được ($P_E > 10^{-3}$).



Hình 15.5: Mức tín hiệu vào máy thu giả định và mô hình hai trạng thái



Hình 15.6: Mô hình Markov hai trạng thái

Vì vậy, trong mô hình hai trạng thái đơn giản được cho ở hình 15.5, các trạng thái có thể được thể hiện bằng tập:

$$S = \{g, b\} \tag{15.5}$$

Tiến triển theo thời gian, kênh chuyển từ trạng thái tốt sang trạng thái xấu và ngược lại. Tốc độ chuyển dịch và độ lâu của mỗi trạng thái phụ thuộc vào tương quan thời gian của quá trình pha đinh (thời gian nhất quán của kênh). Nếu thời gian được đo trong các *số gia của thời gian ký hiệu* (bit), thì ta có thể xây dựng mô hình kênh rời rạc như sau:

Tại thời điểm đầu của mỗi ký hiệu (bit), kênh thuộc một trong hai trạng thái. Nếu kênh ở trạng thái tốt, thì xác suất lỗi truyền dẫn là không đáng kể. Ngược lại, nếu kênh ở trạng thái xấu, thì xác suất lỗi truyền dẫn là lớn không thể chấp nhận được. Trước khi truyền một bit mới, kênh có thể chuyển trạng thái hoặc vẫn giữ nguyên trạng thái hiện tại. Việc chuyển đổi trạng thái diễn ra theo một tập các *xác suất chuyển dịch trạng* thái a_{ij}. Tồn tại các xác suất có điều kiện, và với mô hình hai trạng thái có bốn xác. Ký hiệu S_t và S_{t+1} là trạng thái của kênh tại thời điểm *t* và *t+1* (*lưu ý rằng t & t+1 là chỉ số thời gian rời rạc*). Với việc ký hiệu này, ta định nghĩa 4 xác suất chuyển dịch trạng thái là:

$$\begin{aligned} a_{gg}(t) &= \Pr\{S_{t+1} = g \mid S_t = g\} \\ a_{gb}(t) &= \Pr\{S_{t+1} = b \mid S_t = g\} \\ a_{bg}(t) &= \Pr\{S_{t+1} = g \mid S_t = b\} \\ a_{bb}(t) &= \Pr\{S_{t+1} = b \mid S_t = b\} \end{aligned} \tag{15.6}$$

Phương trình trên được biểu diễn ở dạng ma trận chuyển dịch trạng thái sau:

$$\underbrace{A(t)}_{\text{Ma trận chuyển dịch trạng thái}} = \underbrace{\begin{bmatrix} a_{gg}(t) & a_{gb}(t) \\ a_{bg}(t) & a_{bb}(t) \end{bmatrix}}_{\text{Tổng các phần tử trên một hàng =1}} \tag{15.7}$$

Lưu ý rằng, vì ta đang xét mô hình hai trạng thái, nên giả sử $S_t = g$ thì ta phải có $S_{t+1} = g$ (kênh vẫn ở trạng thái tốt) hoặc $S_{t+1} = b$ (kênh *chuyển sang* trạng thái xấu). Vì vậy, mỗi hàng của ma trận trạng thái phải có tổng bằng 1. Thể hiện ở dạng lưu đồ chuyển dịch trạng thái của mô hình hai trạng thái được cho ở hình 15.6.

Tiếp theo, giả sử mô hình kênh là *dừng,* theo đó ma trận chuyển dịch trạng thái *A(t)* được cố định vì vậy *A(t) = A*. Tuy nhiên, nếu mô phỏng được thực hiện với phân bố xác suất trạng thái khởi đầu $\prod_0$ khác với phân bố trạng thái *bền vững*, đôi khi cần có phân bố trạng thái để tiến đến giá trị trạng thái bền vững $\prod_{ss}$. Cần phải lưu ý đến xác suất tìm kiếm mô hình trong trạng thái cho trước. Định nghĩa $\prod_t$ là phân bố xác suất trạng thái tại thời điểm t. Cụ thể là:

$$\underbrace{\prod_t}_{\substack{\text{Phân bố xác suất} \\ \text{trạng thái tại thời điểm } t}} = \underbrace{\begin{bmatrix} \pi_{t,g} & \pi_{t,b} \end{bmatrix}}_{\substack{\text{Các xác suất tìm kiếm kênh} \\ \text{ở trạng thái tốt hoặc xấu} \\ \text{tại thời điểm } t}} \tag{15.8}$$

Trong đó $\pi_{t,g}$ và $\pi_{t,b}$ biểu diễn các xác suất tìm kiếm kênh trong trạng thái tốt hoặc xấu tại thời điểm *t*. Theo định nghĩa ma trận chuyển dịch trạng thái, phân bố trạng thái tại thời điểm *t + 1* được cho bởi:

$$\prod_{t+1} = \prod_t A \tag{15.9}$$

Theo cách tương tự, phân bố trạng thái tại thời điểm *t + 2* là:

$$\prod_{t+2} = \prod_{t+1} A = \left(\prod_t A\right) A = \prod_t A^2 \tag{15.10}$$

Vì vậy, ở dạng tổng quát:

$$\underbrace{\prod_{t+k}}_{\substack{\text{Phân bố trạng thái} \\ \text{tại thời điểm } t+k}} = \underbrace{\prod_t}_{\substack{\text{Phân bố trạng thái} \\ \text{tại thời điểm } t}} \cdot \underbrace{A^k}_{\substack{\text{Ma trận chuyển dịch} \\ \text{trạng thái tại bước thứ } k}} \tag{15.11}$$

Trong đó A^k thể hiện ma trận chuyển dịch trạng thái bước thứ *k*.

Hầu hết nhưng không phải là tất cả, các quá trình Markov chuyển dần về phân bố xác suất trạng thái *bền vững* theo sự tiến triển thời gian. Giả sử quá trình Markov hội tụ về phân bố giá trị trạng thái bền $\prod_{t+k} = \prod_t$ khi *t* đủ lớn và *k* bất kỳ, theo đó:

$$\prod_{ss} = \prod_{ss} A^k = \begin{bmatrix} \pi_g & \pi_b \end{bmatrix} \tag{15.12}$$

Với *k* bất kỳ. Dễ dàng nhận thấy rằng, khi giá trị k đủ lớn, thì các hàng của ma trận chuyển dịch trạng thái tại bước thứ *k* A^k tạo ra giá trị trạng thái bền vững $\prod_{ss}$. Tính hội tụ của $\prod_0$ về $\prod_{ss}$ được minh họa trong ví dụ dưới đây:

Ví dụ 15.1: Cho ma trận chuyển dịch trạng thái sau:

$$A = \begin{bmatrix} 0,98 & 0,02 \\ 0,05 & 0,95 \end{bmatrix} \tag{15.13}$$

Với phân bố trạng thái khởi đầu là:

$$\prod_0 = \begin{bmatrix} 0,50 & 0,50 \end{bmatrix} \tag{15.14}$$

Chương trình **Matlab NVD15_MMtransient.m** minh họa quá trình hội tụ của $\prod_0$ về $\prod_{ss}$.

Chạy chương trình nhận được kết quả:

Cac xac suat trang thai ben la: 0,71412 và 0,28588.

Gia tri cua ma tran chuyen dich trang thai A^N la:

ans =

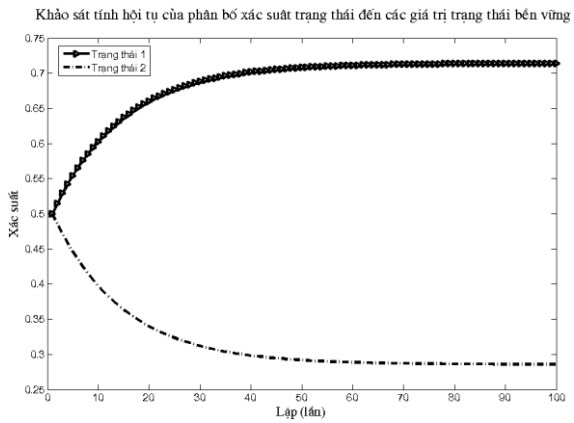
0,7145 0,2855

0,7138 0,2862

Thấy rõ, việc tính toán giá trị của $\prod_{ss}$ lặp lại dùng (15.9) và bằng cách tăng A tới một giá trị lớn để có được các kết quả phù hợp như mong muốn. Cách thức phân bố xác suất hội tụ đến giá trị trạng thái bền được thấy trên hình 15.7.

Trước khi rời khỏi mô hình hai trạng thái, cần phải đề cập ma trận tạo lỗi, được định nghĩa là:

$$B = \begin{bmatrix} \Pr\{C \mid g\} & \Pr\{C \mid b\} \\ \Pr\{E \mid g\} & \Pr\{E \mid b\} \end{bmatrix} \tag{15.15}$$



Hình 15.7: Quá trình hội tụ của phân bố xác suất trạng thái đến các giá trị trạng thái bền

Trong đó "*C*" và "*E*" biểu thị việc thực hiện quyết định đúng & sai tương ứng. Bằng phép nhân ma trận, theo đó xác suất quyết định đúng và sai không điều kiện P_C và P_E được cho bởi:

$$\begin{bmatrix} P_C & P_E \end{bmatrix} = \prod_{SS} B^T \tag{15.16}$$

Trong đó $\prod_{ss}$ là ma trận phân bố trạng thái bền và B^T là chuyển vị của ma trận tạo lỗi *B*.

Lưu ý rằng, nếu tất cả các phần tử của B khác không, thì trạng thái có thể tạo ra một mỗi mặc dù các xác suất lỗi có thể khác khá nhiều đối với các trạng thái khác nhau. Vì vậy, khi quan sát lỗi, trạng thái tạo ra lỗi không thể nhận biết được. Đó là lý do mà ta gọi là mô hình "*ẩn*". Những mô hình này được cho là những mô hình Markov ẩn (HMM).

Ví dụ 15.2: Trong ví dụ này, mô phỏng kênh dùng mô hình Markov được sáng tỏ và thực hiện tính toán xác suất lỗi hệ thống. Giả sử các lỗi được tạo ra trong trạng thái, hiển nhiên xác suất lỗi trong trạng thái tốt $\Pr\{E \mid g\}$ sẽ nhỏ hơn nhiều so với xác suất lỗi trong trạng thái xấu $\Pr\{E \mid b\}$. Cụ thể, ta xác định các xác suất lỗi có điều kiện:

$$\Pr\{E \mid g\} = 0{,}0005 \tag{15.17}$$

$$\Pr\{E \mid b\} = 0{,}1000 \tag{15.18}$$

Với các xác suất lỗi này, ma trận tạo lỗi *B* có dạng:

$$B = \underbrace{\begin{bmatrix} 0{,}9995 & 0{,}9000 \\ 0{,}0005 & 0{,}1000 \end{bmatrix}}_{\text{Tổng các phần tử trên một cột =1}} \tag{15.19}$$

Ngoài ra, chuỗi Markov được định nghĩa bởi ma trận chuyển dịch trạng thái:

$$A = \underbrace{\begin{bmatrix} 0{,}98 & 0{,}02 \\ 0{,}05 & 0{,}95 \end{bmatrix}}_{\text{Tổng các phần tử trên mỗi hàng =1}} \tag{15.20}$$

Như trong ví dụ trước.

Chương trình Matlab để mô phỏng kênh được cho bởi **NVD15_hmm2.m** trong Phụ lục 5A.

Kết quả chạy chương trình mô phỏng **NVD15_hmm2.m** cho bài toán này là: P_E = 0,0285.

Vì vậy, ta có:

$$P_E = 0{,}0285 \tag{15.21}$$

Lưu ý rằng, P_E được tính toán theo kiểu này là biến ngẫu nhiên. Bây giờ ta "*kiểm tra-tính hợp lý*" kết quả mô phỏng này.

Từ ví dụ trước, các xác suất trạng thái bền vững là:

$$\prod_{ss} = \begin{bmatrix} 0{,}7141 & 0{,}2859 \end{bmatrix} \tag{15.22}$$

Từ (15.16) ta có:

$$\begin{bmatrix} P_C & P_E \end{bmatrix} = \begin{bmatrix} 0,7141 & 0,2859 \end{bmatrix} \begin{bmatrix} 0,9995 & 0,0005 \\ 0,9000 & 0,1000 \end{bmatrix} \tag{15.23}$$

Cho ta:

$$\begin{bmatrix} P_C & P_E \end{bmatrix} = \begin{bmatrix} 0,9711 & 0,0289 \end{bmatrix} \tag{15.24}$$

Nó $P_E = 0,0289$ rất sát với giá trị được cho bởi mô phỏng $P_E = 0,0285$. Vì vậy, kết quả mô phỏng là hợp lý.

Phiên bản tổng quát hơn của mô hình Markov hai trạng thái được xây dựng với một số lượng lớn các trạng thái, với nhiều trạng thái tốt và xấu với mức độ thay đổi về "*tính tốt*" hoặc "*tính xấu*" tương ứng với sự tăng dần mức độ pha đinh và/hoặc tạp âm. Kết quả là mô hình Markov *N* trạng thái, là sự tổng quát hóa của mô hình hai trạng thái.

15.3.2. Mô hình Markov N trạng thái

Một mô hình (quá trình) Markov N trạng thái cho kênh truyền thông rời rạc được định nghĩa bởi nhiều tham số. Tập các trạng thái S được biểu thị bằng:

$$S = \{1, 2, 3, ..., N\} \tag{15.25}$$

Và giống như trên, S_t biểu thị cho trạng thái tại thời điểm *t*. Vì vậy, S_t đi khắp tập *S*. Ta quan tâm xác suất $\prod_{t,i}$ là xác suất mà mô hình Markov sẽ ở trạng thái *i* tại thời điểm *t*, nghĩa là:

$$\prod_{t,i} = \Pr[S_t = i], \qquad i = 1,2,...,N \tag{15.26}$$

Các xác suất chuyển dịch trạng thái a_{ij} biểu thị xác suất chuyển từ trạng thái *i* tại thời điểm *t* ($S_t = i$) sang trạng thái *j* tại thời điểm *t + 1* ($S_{t+1} = j$). Ở dạng phương trình:

$$a_{ij} = \underbrace{\Pr[S_{t+1} = j \mid S_t = i]}_{\substack{\text{Xác suất chuyển dịch trạng thái: từ trạng thái} \\ i \text{ ở thời điểm } t \text{ sang trạng thái } j \text{ ở thời điểm } t+1}}, \quad i, j = 1,2,...,N \tag{15.27}$$

Các chuyển dịch này thường diễn ra trong một số gia thời gian bằng thời gian một bit hoặc một ký hiệu. Cuối cùng, ta có các xác suất (ký hiệu tin bị lỗi) chuyển dịch từ đầu vào tới đầu ra. Các ký hiệu lỗi đối với bảng mẫu tự ký hiệu *M-ary* được biểu thị bởi tập:

$$E = \{e_1, e_2, ..., e_M\} \tag{15.28}$$

Với trường hợp nhị phân quen thuộc thì:

$$E = \{0,1\} \tag{15.29}$$

Trong đó 0 biểu thị không lỗi và 1 biểu thị lỗi. Xác suất xuất hiện ký hiệu lỗi e_k với giả thiết mô hình đang ở trạng thái *i* được biểu thị là $b_i(e_k)$. Ở dạng phương trình ta có:

$$\underbrace{b_i(e_k) = \Pr[e_k \mid S_t = i]}_{\substack{\text{Xác suất xảy ra ký hiệu lỗi thứ } k, (e_k), \\ \text{với giả thiết mô hình đang} \\ \text{ở trạng thái thứ } i \text{ tại thời điểm } t}} \tag{15.30}$$

Trong các ứng dụng quyết định cứng nhị phân, thì các tham số $b_i(e_k)$ được biểu diễn bởi một ma trận 2 hàng và *N* cột, trong đó *N thể hiện số trạng thái trong mô hình*. Vì vậy, ma trận tạo lỗi *B* có dạng:

$$B = \underbrace{\underbrace{\begin{bmatrix} \overbrace{b_{11} \quad b_{12} \quad \ldots \quad b_{1i} \quad \ldots \quad b_{1N}}^{\text{Các xác suất quyết định đúng}} \\ \underbrace{b_{21} \quad b_{22} \quad \ldots \quad b_{2i} \quad \ldots \quad b_{2N}}_{\text{Các xác suất lỗi}} \end{bmatrix}}_{\text{Tổng các phần tử trên mỗi cột =1}}}_{\substack{\text{Trường hợp không phải là nhị phân hoặc quyết định mềm} \\ \text{thì ma trận tạo lỗi B có nhiều hơn 2 hàng}}} \tag{15.31}$$

Trong đó b_{1i} biểu thị xác suất quyết định đúng với điều kiện mô hình ở trạng thái *i,* và b_{21} thể hiện xác suất lỗi với điều kiện hệ thống ở trạng thái *i*. Với phương trình này, lưu ý rằng các cột của *B* phải có tổng bằng 1. Với các ứng dụng phi nhị phân hoặc các ứng dụng *quyết định mềm* nhị phân, thì ma trận tạo lỗi B sẽ có *nhiều hơn hai hàng*.

Những tham số này định nghĩa quá trình (mô hình) Markov rời rạc theo thời gian hoạt động tại tốc độ chuyển dịch bằng tốc độ ký hiệu của hệ thống truyền thông, và đầu ra của quá trình bao gồm *hai chuỗi*: chuỗi các trạng thái $\{S_t\}$ và chuỗi các ký hiệu lỗi $\{E_t\}$, trong đó *t* là *chỉ số thời gian* được đánh chỉ số trên tập số nguyên $\{0,1,2....\}$. Thông thường, chỉ có đầu vào và đầu ra của kênh và vì thế chuỗi lỗi có thể quan sát được. Bản thân chuỗi trạng thái không thể quan sát được. Vì vậy, chuỗi trạng thái là **"*ẩn*"** hoặc không nhìn thấy từ các quan sát bên ngoài và mô hình Markov được gọi là mô hình Markov *ẩn*.

15.3.3. Quá trình Markov bậc một

Quá trình Markov bậc *m* (nhớ *m*), được định nghĩa là xác suất của trạng thái tại thời điểm $t + 1$, S_{t+1} được cho bởi:

$$\Pr\left[S_{t+1} \middle| S_t, S_{t-1}, \ldots\right] = \Pr\left[S_{t+1} \middle| S_t, S_{t-1}, \ldots, S_{t-m-1}\right] \tag{15.32}$$

Cho thấy xác suất phụ thuộc vào *m* trạng thái trước đó.

Với quá trình Markov bậc nhất, thì xác suất của trạng thái tại thời điểm t + 1 chỉ là một hàm hàm của trạng thái *trước đó*. Nói cách khác:

$$P_r\left[S_{t+1} \mid S_t, S_{t-1}, \ldots\right] = \Pr\left[S_{t+1} \mid S_t\right] \tag{15.33}$$

Đối với các mô hình kênh, ta sẽ *dùng một mô hình Markov bậc nhất.* Đối với việc mô hình hóa nguồn số liệu, có thể cần đến các mô hình bậc cao hơn. Ví dụ, mô hình hóa các chuỗi ký hiệu trong văn bản Anh ngữ có thể cần đến mô hình Markov bậc 2 hoặc bậc 3, vì xác suất xuất hiện của một chữ cái cho trước có thể phụ thuộc vào hai hoặc ba chữ cái trước đó.

15.3.4. Tính dừng

Vì tính dừng thường được giả định trong quá trình mô hình hóa các phần tương tự của kênh truyền thông, nên cũng thường giả định tính dừng của mô hình Markov đối với các kênh rời rạc. Tính dừng để đề cập các tham số của mô hình, nghĩa là các xác suất $\prod_{t,i}$, $a_{ij}(t)$ và $b_i(e_k)$ không phụ thuộc vào t. Trong trường hợp này, ta có:

$$\underbrace{\Pr[S_{t+1}=i]}_{\substack{\text{Xác suất mô hình ở}\\ \text{trạng thái i}\\ \text{tại thời điểm t+1}}} = \prod_i = \sum_{k=1}^{N} \underbrace{\Pr[S_t=k]}_{\substack{\text{xác suất mô hình}\\ \text{ở trạng thái k}\\ \text{tại thời điểm t}\\ \text{(xác suất trạng thái k)}}} \cdot \underbrace{\Pr[S_{t+1}=i \mid S_t=k]}_{\substack{\text{xác suất mô hình chuyển từ}\\ \text{trạng thái k tại ở thời điểm t}\\ \text{sang trạng thái i tại thời điểm t+1}}} \tag{15.34}$$

$$= \sum_{k=1}^{N} \pi_k a_{ki}, \qquad i = 1,...,N$$

Trong đó các số hạng a_{ki} là các phần tử trong cột thứ i của *ma trận chuyển dịch trạng thái:*

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1i} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2i} & \dots & a_{2N} \\ \dots & \dots & \ddots & \dots & \ddots & \dots \\ a_{N1} & a_{N2} & \dots & a_{Ni} & \dots & a_{NN} \end{bmatrix} \tag{15.35}$$

Và π_k là các phần tử của véc tơ phân bố xác suất trạng thái:

$$\prod = [\pi_1 \quad \pi_2 \quad ... \quad \pi_N] \tag{15.36}$$

Như trong trường hợp mô hình hai trạng thái, ma trận chuyển dịch *n tầng* được cho bởi A^n. Như đã giải thích trong ví dụ 15.1 đối với mô hình 2 trạng thái, (15.34) cùng với ràng buộc:

$$\sum_{i=1}^{N} \pi_i = 1 \tag{15.37}$$

Hàm ý rằng các xác suất trạng thái π_i được xác định *duy nhất* từ các xác suất chuyển dịch trạng thái a_{ij}. *Vì vậy*, mô hình Markov hoàn toàn được định nghĩa bởi: *(i)* Ma trận xác suất chuyển trạng thái A (các phần tử của A là các xác suất chuyển dịch trạng thái của mô hình); *(ii)* Ma trận tạo lỗi B (ma trận xác suất chuyển dịch ký hiệu từ đầu vào tới đầu ra B, các phần tử của B là các xác suất chuyển dịch ký hiệu từ đầu vào tới đầu ra):

$$B = \begin{bmatrix} b_{11} & \cdots & b_{1N} \\ \vdots & \ddots & \vdots \\ b_{M1} & \cdots & b_{MN} \end{bmatrix} \tag{15.38}$$

Như đã thảo luận trước.

15.3.5. Mô phỏng mô hình Markov

Một khi rút ra được mô hình Markov rời rạc, thì việc mô phỏng mô hình là khá dễ dàng. Thấy rõ ở ví dụ 15.2 cho mô hình 2 trạng thái, và kỹ thuật dùng trong ví dụ 15.2 giờ đây được tổng quát hóa cho mô hình N trạng thái.

Giả sử rằng mô hình đang ở trạng thái k tại thời điểm t, và cả trạng thái tiếp theo và thành phần của véc tơ lỗi e_{k+1} tương ứng đều được mô phỏng. Mô phỏng được tiến hành theo hai bước cơ bản sau:

Bước 1 *(xác định xác suất chuyển dịch trạng thái a_{ki} nghĩa là thiết lập trạng thái mới)*: Trước hết là tạo hai biến ngẫu nhiên phân bố đều U_1 và U_2. Biến ngẫu nhiên thứ nhất U_1 được dùng để xác định trạng thái mới. Giả sử, mô hình đang ở trạng thái k, thì các xác suất chuyển dịch được xác định bởi hàng thứ k của ma trận chuyển dịch trạng thái A. Hàng thứ k được định rõ bởi véc tơ A_k:

$$A_k = \begin{bmatrix} a_{k1} & a_{k2} & \dots & a_{ki} & a_{k,i+1} & \dots & a_{kN} \end{bmatrix} \tag{15.39}$$

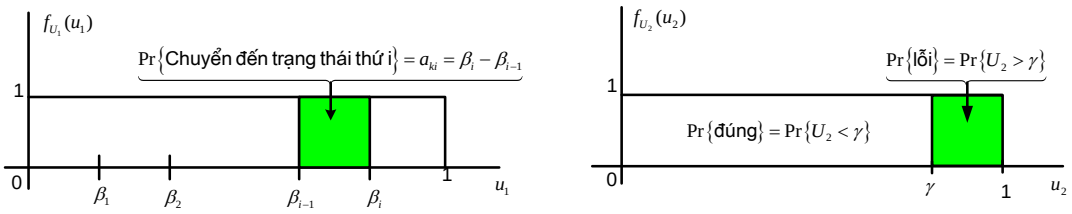
Các xác suất tích lũy liên kết với A_k được ký hiệu là β_k và được cho bởi:

$$\beta_k = \sum_{j=1}^{k} a_{kj} \tag{15.40}$$

Lưu ý rằng sẵn có hàm **cumsum** trong thư viện Matlab thực hiện xắp sếp véc tơ A_k vào một véc tơ có các phần tử β_k. Xác suất tạo sự chuyển dịch từ trạng thái k sang trạng thái i theo định nghĩa là a_{ki}, được cho bởi:

$$a_{ki} = \beta_i - \beta_{i-1} \tag{15.41}$$

Xác suất này được biểu diễn bởi vùng đậm trong hình 15.8(a).



a) Xác định xác suất chuyển dịch trạng thái (bước 1) b) Xác định xác suất lỗi (bước 2)

Hình 15.8: Các bước mô phỏng mô hình Markov

Bước 2 *(xác định xác suất lỗi ở trạng thái mới)*: Sau khi trạng thái mới được thiết lập, bước tiếp theo là xác định lỗi có xảy ra ở trạng thái mới không. Muốn vậy, ta so sánh biến ngẫu nhiên phân bố đều thứ hai U_2 với ngưỡng γ được cho ở hình 15.8(b). Xác suất quyết định đúng là $\Pr\{U_2 < \gamma\}$, và xác suất lỗi là $\Pr\{U_2 > \gamma\}$ là vùng đậm trong hình 15.8(b). Vì trạng thái hiện tại là i, nên giá trị của γ được cho bởi phần tử b_{1i} (hàng 1, cột i) trong ma trận quan sát trạng thái B.

Mã chương trình Matlab để thực hiện thủ tục mô phỏng cho bài toán này được cho bởi **NVD15_ext1.m** trong Phụ lục 15A.

Bây giờ ta sẽ giải thích quá trình này trong ví dụ dưới đây.

Ví dụ 15.3: *[Mô hình ba trạng thái quyết định cứng]:* Trong ví dụ này, một chuỗi nhị phân có chiều dài L = 20000 được tạo ra để thể hiện các lỗi ký hiệu trên kênh. Chuỗi lỗi này

được tạo ra bằng cách dùng mô hình Markov ba trạng thái. Nếu cho trước một phần tử trong chuỗi đó là số "1" nhị phân, thì một lỗi ký hiệu kênh được ghi vào vị trí tương ứng với phần tử đó. Một số "0" nhị phân biểu thị rằng không xảy ra lỗi tại vị trí đó.

Ta giả thiết rằng các lỗi có thể xảy ra trong bất kỳ trạng thái nào thuộc ba trạng thái đó. Cụ thể, ta giả thiết:

$$\Pr\{E \mid S_1\} = 0{,}0010 \tag{15.42}$$

$$\Pr\{E \mid S_2\} = 0{,}0500 \tag{15.43}$$

$$\Pr\{E \mid S_3\} = 0{,}0100 \tag{15.44}$$

Tương ứng với ma trận xác suất lỗi (ma trận tạo lỗi) là:

$$B = \underbrace{\left[\begin{array}{ccc} \overbrace{0{,}9990 & 0{,}9500 & 0{,}9900}^{\text{Các xác suất quyết định đúng}} \\ \underbrace{0{,}0010 & 0{,}0500 & 0{,}0100}_{\text{Các xác suất quyết định sai}} \end{array}\right]}_{\text{Tổng các phần tử trên mỗi cột bằng 1}}_{\text{(quyết định cứng =2) x (số trạng thái N=3)}} \tag{15.45}$$

$$\underbrace{\qquad\qquad}_{\substack{\text{Ma trận các xác suất (lỗi) chuyển dịch ký hiệu vào ra:} \\ \text{Số hàng của B thể hiện quyết định cứng hoặc mềm} \\ \text{Số cột của B thể hiện số trạng thái trong mô hình}}}$$

Ma trận chuyển dịch trạng thái đối với mô hình Markov ba trạng thái được giả thiết là:

$$A = \underbrace{\left[\begin{array}{ccc} 0{,}80 & 0{,}10 & 0{,}10 \\ 0{,}20 & 0{,}60 & 0{,}20 \\ 0{,}02 & 0{,}08 & 0{,}90 \end{array}\right]_{3x3}}_{\sum_{j=1}^{3} a_{i,j} = 1,\ (\text{tổng số phần tử trên hàng} = 1)} \tag{15.46}$$

Ta giả sử mô hình này khởi đầu ở trạng thái 1.

Mã chương trình Matlab **NVD15_errvector.m** tạo chuỗi lỗi được cho ở Phụ lục 15A. Khi chạy chương trình Matlab **NVD15_errvector.m** phải nhập các tham số đầu vào là *L*, ma trận *B* và *A*. Các giá trị mặc định là *L* = 20000 và các ma trận *B* và *A* lần lượt được định nghĩa bởi (15.45) và (15.46). Chương trình tạo ra hai đầu ra: ***(i)*** véc tơ lỗi **out** cho biết các vị trí lỗi trong chuỗi của *L* lần truyền dẫn; ***(ii)*** véc tơ chuỗi trạng thái **state_seq** trình bày chi tiết các chuyển dịch trạng thái. Có thể tính toán: ***(i)*** xác suất tìm kiếm mô hình ở một trạng thái cho trước từ véc tơ chuỗi trạng thái; ***(ii)*** xác suất lỗi được mô phỏng từ véc tơ lỗi. Các tính toán này được thực hiện trong mã chương trình Matlab **NVD15rhmmtest.m**. Chạy chương trình này với các giá trị mặc định tạo ra các đầu ra dưới đây, lưu ý rằng các hội thoại (yêu cầu đầu vào của chương trình) liên quan đến việc loại bỏ hoặc chấp nhận giá trị mặc định được ẩn.

```
>> NVD15_hmmtest

    Các kết quả mô phỏng:
    Xác suất trạng thái 1 là:        0,2432
```

Xác suất trạng thái 2 là: 0,1634
Xác suất trạng thái 3 là: 0,5934
Xác suất lỗi là: 0,01445

Lưu ý rằng, các giá trị mô phỏng được cho đối với các xác suất trạng thái và các xác suất lỗi là các biến ngẫu nhiên (giá trị ngẫu nhiên). Giảm phương sai của các biến ngẫu nhiên này bằng cách tăng độ dài véc tơ lỗi.

Có thể "*kiểm tra tính hợp lý*" đối với các giá trị trên bằng cách tính toán các xác suất trạng thái và xác suất lỗi trực tiếp từ hai ma trận *B* và *A*. Mã chương trình Matlab cùng với kết quả đầu ra như sau (lưu ý rằng, phải nhập các ma trận A và B trước khi thực hiện các dòng lệnh này):

```
>> A100=A^100
```

$$A100 = \begin{bmatrix} 0,2353 & 0,1765 & 0,5882 \\ 0,2353 & 0,1765 & 0,5882 \\ 0,2353 & 0,1765 & 0,5882 \end{bmatrix}$$

```
>> PE = B * A100'
```

$$PE = \begin{bmatrix} 0,9851 & 0,9851 & 0,9851 \\ 0,0149 & 0,0149 & 0,0149 \end{bmatrix}$$

Lưu ý rằng, xác suất trạng thái lý thuyết S_1, S_2, S_3 lần lượt là 0,2353; 0,1765; 0,5882 và xác suất lỗi lý thuyết là 0,0149. Thấy rõ, các kết quả này hơi khác so với các kết quả mô phỏng, như đã đề cập ở trên, là các biến (giá trị) ngẫu nhiên. Phương sai của các biến ngẫu nhiên này có thể được giảm như đã đề cập ở trên, để được tường minh ta nên kiểm tra bằng cách chọn giá trị L lớn hơn.

15.4. Mô hình Gillbert và Fritchman

Đặc tính hóa các kênh rời rạc dùng các chuỗi Markov hữu hạn trạng thái, rời rạc theo thời gian đã được đề xuất bởi Gillbert, Fritchman và nhiều tác giả khác.

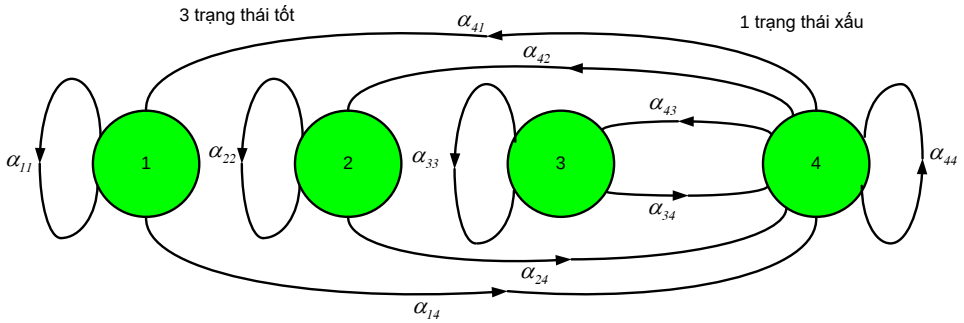
Mô hình Gillbert là một mô hình hai trạng thái có một trạng thái tốt không lỗi, và một trạng thái xấu với một xác suất lỗi p. Mô hình này đã được Gillbert dùng để tính dung lượng của kênh có các lỗi cụm. Các tham số của mô hình có thể được ước tính từ dữ liệu đo hoặc mô phỏng bằng các thủ tục sẽ được khảo sát trong các phần sau đây.

Mô hình Fritchman lần đầu tiên được đề xuất vào năm 1967, hiện đang được quan tâm nhiều, vì mô hình Fritchman tỏ ra phù hợp với việc mô hình hóa các *lỗi cụm* trong các kênh vô tuyến di động và cũng vì nó dễ dàng ước tính các tham số của mô hình Fritchman từ các phân bố lỗi cụm. Đối với các kênh nhị phân, cốt lõi của Fritchman là phân chia không gian trạng thái thành *k* trạng thái tốt và *N-k* trạng thái xấu. Các trạng thái tốt thể hiện các truyền dẫn không lỗi và các trạng thái xấu biểu thị lỗi truyền dẫn. Vì vậy, các thực thể trong ma trận B là 0 và 1 và không nhất thiết phải ước tính chúng. Mô hình Fritchman với ba trạng thái tốt và một trạng thái xấu được minh họa trong hình 15.9. Mô hình này có ma trận chuyển dịch trạng thái:

$$A = \begin{bmatrix} a_{11} & 0 & 0 & a_{14} \\ 0 & a_{22} & 0 & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \tag{15.47}$$

Ma trận B cho trường hợp này có dạng đơn giản là:

$$B = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{15.48}$$



Hình 15.9: Mô hình Fritchman có ba trạng thái tốt và một trạng thái xấu

Tổng quát, ma trận chuyển dịch trạng thái *A* cho mô hình Fritchman có thể được phân chia hóa như sau:

$$A = \begin{bmatrix} A_{gg} & A_{gb} \\ A_{bg} & A_{bb} \end{bmatrix} \tag{15.49}$$

Trong đó các ma trận con biểu diễn các xác suất chuyển dịch giữa các trạng thái tốt và xấu. Với mô hình này, có thể rút ra biểu thức giải tích cho các phân bố lỗi cụm theo các tham số mô hình và dùng các biểu thức này để ước tính các tham số của mô hình từ các phân bố lỗi cụm kinh nghiệm (đo đạc hoặc mô phỏng).

Đặt $\bar{O} = \{O_1, O_2, ..., O_T\}$ là một chuỗi lỗi với $O_k = 1$ biểu thị bit phát thứ *k* phải chịu một lỗi truyền dẫn, và $O_k = 0$ biểu thị cho truyền dẫn bit thứ *k* không bị lỗi. Ngoài ra, ký hiệu $\{0^m \mid 1\}$ biểu thị cho sự kiện quan sát được *m* hoặc nhiều hơn *m* lần truyền dẫn không lỗi liên tiếp theo sau xảy ra một lỗi, và $\{1^m \mid 0\}$ biểu thị cho sự kiện quan sát *m* hoặc nhiều hơn *m* lỗi liên tiếp theo sau một lần truyền không lỗi. Fritchman chỉ ra rằng, các xác suất xảy ra hai sự kiện này được cho bởi tổng của các hàm mũ được đánh trọng số dưới đây:

$$\Pr\{0^m \mid 1\} = \sum_{i=1}^{k} f_i \lambda_i^{m-1} \tag{15.50}$$

$$\Pr\{1^m \mid 0\} = \sum_{i=k+1}^{N} f_i \lambda_i^{m-1} \tag{15.51}$$

Trong đó λ_i , i = 1, 2, ..., k và λ_i , i = k + 1, k + 2, ..., N lần lượt là các giá trị riêng của A_{gg} và A_{bb}, và tương ứng với giá trị của f_i là các hàm của a_{ij}. Từ (15.50) và (15.51) ta có thể có được xác suất tìm được chính xác m số 0 (hoặc số 1) như sau:

$$\Pr\left(0^{m-1}\,|\,1\right)-\Pr\left(0^{m}\,|\,1\right)=\sum_{i=1}^{k} f_i\lambda_i^{m-2}\left(1-\lambda_i\right) \tag{15.52}$$

$$\Pr\left(1^{m-1}\,|\,0\right)-\Pr\left(1^{m}\,|\,0\right)=\sum_{i=k+1}^{N} f_i\lambda_i^{m-2}\left(1-\lambda_i\right) \tag{15.53}$$

Mô hình Fritchman có thể được hiểu là tương đương với một quá trình Markov có một ma trận xác suất chuyển dịch trạng thái:

$$\tilde{A}=\begin{bmatrix}\Lambda_{gg} & A_{gb}\\ A_{bg} & \Lambda_{gg}\end{bmatrix} \tag{15.54}$$

Trong đó Λ_{gg} $v\mu$ Λ_{bb} là các ma trận đường chéo được cho bởi:

$$\Lambda_{gg}=\begin{bmatrix}\lambda_1 & 0 & \cdots & 0\\ 0 & \lambda_2 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & \lambda_k\end{bmatrix} \tag{15.55}$$

$$\Lambda_{bb}=\begin{bmatrix}\lambda_{k+1} & 0 & \cdots & 0\\ 0 & \lambda_{k+2} & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & \lambda_N\end{bmatrix} \tag{15.56}$$

Lưu ý rằng trong mô hình tương đương này, không có các chuyển dịch trong tập của k trạng thái tốt và không có các chuyển dịch trong tập của N - k trạng thái xấu. Các chuyển dịch đó không thể phân biệt được từ chuỗi lỗi quan sát, vì quá trình chuyển dịch của một trạng thái tốt này sang một trạng thái tốt khác vẫn không tạo ra các lỗi và quá trình chuyển dịch không thể quan sát được từ đầu ra. Cấu trúc này được nhắc lại khi xét các *mô hình bán Markov* ở phần sau.

Mô hình Fritchman không duy nhất ngoại trừ khi chỉ có một trạng thái xấu. Trường hợp tổng quát, đây cũng là lý do mà phân bố liên tiếp không lỗi $\Pr\{0^m\,|\,1\}$ và phân bố lỗi cụm (liên tiếp m lỗi) $\Pr\{1^m\,|\,0\}$ không định rõ tính phụ thuộc thống kê của các cụm lỗi và liên tục không lỗi. Trong trường hợp mô hình trạng thái đơn lỗi như hình 15.9, Fritchman chỉ ra rằng:

$$\Pr\left(0^{m}\,|\,1\right)=\sum_{k=1}^{N-1}\frac{a_{Nk}.\left(a_{kk}\right)^{m}}{a_{kk}} \tag{15.57}$$

Từ (15.57) ta có thể thấy rằng, trường hợp mô hình trạng thái đơn lỗi, chuỗi không lỗi định rõ duy nhất 2(N - 1) tham số mô hình. Dùng thủ tục kinh nghiệm để làm cho phù hợp (N - 1) hỗn hợp hàm mũ như trong (15.50) và (15.51) với phân bố chuỗi không lỗi (được mô phỏng hoặc được đo đạc).

Trong khi mô hình Fritchman có thể ứng dụng cho các kênh rời rạc có các phân bố lỗi cụm đơn giản, thì nó có thể không phù hợp để đặc tính hóa các mẫu lỗi cụm rất phức tạp (cần có nhiều trạng thái lỗi trong mô hình). Trong những trường hợp này, rất khó ước tính các tham số mô hình từ chỉ riêng các phân bố lỗi cụm. May thay, có thể tìm ước tính khả năng giống nhất ML của các tham số bằng cách dùng các kỹ thuật lặp.

15.5. Ước tính các tham số của mô hình Markov

Mô hình Markov cho kênh rời rạc được mô tả bởi một ma trận chuyển dịch trạng thái A kích cỡ $N \times N$ và ma trận tạo xác suất lỗi B kích cỡ $M \times N$. Tìm được một thủ tục lặp để ước tính các tham số $\Gamma = \{A, B\}$ này từ một chuỗi lỗi cho trước thông qua việc mô phỏng hoặc đo đạc $\overline{O} = \{O_1, O_2, ..., O_T\}$ trên cơ sở giải thuật Baum-Welch. Thuật toán lặp này được thiết kế để hội tụ đến bộ ước tính khả năng giống nhất ML của $\Gamma = \{A, B\}$ mà làm cực đại hóa $\Pr(\overline{O} \mid \Gamma)$.

Mục đích là tính toán các ước tính của các phần tử ma trận chuyển dịch trạng thái A. Chúng được cho bởi:

$$\hat{a}_{ij} = \frac{\textit{Số lần chuyển dịch được mong đợi từ trạng thái i đến trạng thái j}}{\textit{Số lần chuyển dịch được mong đợi từ trạng thái i}} \tag{15.58}$$

Ta cũng cần có các ước tính của các phần tử của ma trận tạo lỗi B, được cho bởi:

$$\hat{b}_j(e_k) = \frac{\textit{Số lần được mong đợi } e_k \textit{ được phát từ trạng thái j}}{\textit{Số lần thăm lại được mong đợi đến trạng thái j}} \tag{15.59}$$

Các tính toán cần thiết để thực thi thuật toán Baum-Welch được mô tả trong các bước dưới đây. Ngoài ra, ta cũng triển khai chương trình Matlab để thực hiện thuật toán Baum-Welch. Hai phiên bản mã mã chương trình Matlab thực hiện tính toán chi tiết. *Phiên bản đầu* chỉ rõ "*mã chương trình cơ bản*" chứa nhiều vòng lặp và vì thế nó không hiệu quả. Tuy nhiên, nó dễ dàng liên hệ mã chương trình cơ bản này với các phương trình được định nghĩa. *Phiên bản sau* thể hiện "*hiệu quả hơn*", loại bỏ một vài hoặc tất cả các hoạt động vòng lặp để nhận được các ưu điểm về khả năng thực hiện các phép toán véc tơ của Matlab. Cần lưu ý rằng, trong một vài trường hợp mã chương trình Matlab không được *véc tơ hóa* hoàn toàn vì mục đích để dễ hiểu mã chương trình hơn. Phiên bản của thuật toán Baum-Welch được cho ở phụ lục 15A được triển khai bằng cách kết hợp các đoạn mã được véc tơ hóa.

Bước 0: Bắt đầu bằng một mô hình (giả định) khởi tạo $\Gamma = \{A, B\}$

Bước 1: Với $\Gamma = \{A, B\}$ như một mô hình, trước tiên ta tính toán "*các biến tiến*":

$$\alpha_t(i) = \Pr\left[O_1, O_2, ...O_t, s_t = i \middle| \Gamma\right] \tag{15.60}$$

Và *"các biến lùi"*:

$$\beta_t = \Pr\left[O_{t+1}, O_{t+2}, ..., O_T \middle| s_t = i, \Gamma\right] \tag{15.61}$$

Với $t = 1, 2, ..., T$ và $i = 1, 2, ..., N$.

Chi tiết về các tính toán này như sau:

Các biến tiến: Tính toán các biến tiến bao gồm 3 bước: khởi tạo, quy nạp, kết thúc.

Khởi tạo:

$$\alpha_1(i) = \pi_i b_i(O_1), \quad i = 1,2,...,N \tag{15.62}$$

Quy nạp:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j\left(O_{t+1}\right), \quad 1 \le t \le T-1, \quad 1 \le j \le N \tag{15.63}$$

Kết thúc:

$$\Pr\left[\bar{O} \mid \Gamma\right] = \sum_{i=1}^{N} \alpha_T(i)\beta_T(i) \tag{15.64}$$

Lưu ý rằng

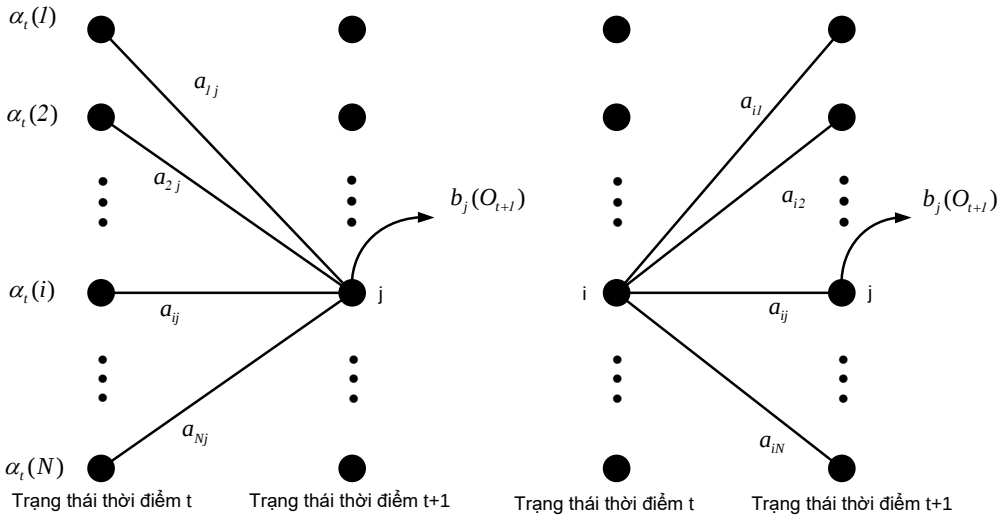
$$\sum_{i=1}^{N} \alpha_T(i) = \sum_{i=1}^{N} \Pr\left[O_1,...,O_T, s_T = i \middle| \Gamma\right] = \Pr\left[\bar{O} \middle| \Gamma\right] \tag{15.65}$$

Mã chương trình Matlab cơ bản để thực hiện các tính toán này như sau: (lưu ý dùng hệ số tỉ lệ. Hệ số tỉ lệ được mô tả trong phần sau)

```
%Phiên bản 1:
alpha               =   zeros(len,state);   % Phân bổ bộ nhớ
for column=1:states
    alpha(1,column) =   pye(1,column)*b(1,column);
    hspace*50em                             % Khởi tạo
end

scale(1)    = sum(alpha(1,:));              % Hệ số chuẩn hóa
alpha(1,:)  = alpha(1,:)/scale(1,:);        % chuẩn hóa
sum1=0;

for t=1:(len-1)                             % Quy nạp
    for j=1:states
        for i=1:states
            inner_sum   = alpha(t,i)*p(i,j);
            sum1        = sum1  +   inner_sum;
        end
        alpha(t+1,j)    = sum1*b(out(t+1)+1,j);
        sum1            = 0;
    end
    scale(t+1)          = sum(alpha(t+1,:));      % Hệ số chuẩn hóa
    alpha(t+1,:)        = alpha(t+1,:)/scale(t+1);% Chuẩn hóa
end
```

Mã chương trình Matlab hiệu quả hơn bằng cách khử một số vòng lặp như đã đề cập ở trên.

```
%Phiên bản 2:
alpha       = zeros(len,states);                    % Phân bổ bộ nhớ
alpha(1,:)  = pye.*b(1,:);                          % Khởi tạo
scale(1)    = sum(alpha(1,:));                      % Hệ số chuẩn hóa
alpha(1,:)  = alpha(1,:)/scale(1);                  % Chuẩn hóa

for t=1:len-1                                       % induction
    alpha(t+1,:)    = (alpha(t,:)*p).*b(out(t+1)+1,:);
    scale(t+1)      = sum(alpha(t+1,:));            % Hế số chuẩn hóa
    alpha(t+1,:)    = alpha(t+1,:)/scale(t+1);      % Chuẩn hóa
end
```



Hình 15.10: Ước tính tham số cho mô hình Markov ẩn (HMM)

Các biến lùi: Tính toán các biến lùi bao gồm 2 bước: khởi tạo và quy nạp

Khởi tạo:

$$\beta_T(i) = 1, \quad i = 1,2,...,N \tag{15.66}$$

Quy nạp:

$$\beta_t(t) = \sum_{j=1}^{N} \beta_{t+1}(j) b_j(O_{t+1}) a_{ij}; \quad 1 \le t \le T-1, \quad 1 \le j \le N \tag{15.67}$$

Các chi tiết về các tính toán này được minh họa trong hình 15.10. Mã chương trình Matlab cơ bản cho đoạn này như sau:

```
%Phiên bản 1:
beta        = zeros(len,states);        % Phân bổ nhớ
beta(len,:) = 1/scale(len);             % Khởi tạo và lấy tỉ lệ
for t=(len-1):-1:1                      % induction
    for i = 1:states
        for j = 1:states
```

```
            inner_sum = p(i,j)...
                *b(out(t+1)+1,j)*beta(t+1,j);
            sum2        = sum2 + inner_sum;
        end
        beta(t,i)   = sum2;
        sum2        = 0;
    end
    beta(t,:)       = beta(t,:)/scale(t); % Lấy tỉ lệ
end
```

Phiên bản hiệu quả hơn (véc tơ hóa từng phần) như sau:

```
%Phiên bản 2:
beta        = zeros(len,states);        % Phân bổ nhớ
beta(len,:) = 1/scale(len);             % Khởi tạo và lấy tỉ lệ

for t=(len-1):-1:1                      % Quy nạp
    beta(t,:)= (beta(t+1,:).*b(out(t+1)+1,:))*(p')/scale(t);
end
```

Bước 2: Bước tiếp theo là tính toán $\gamma_t(i)$ theo

$$\gamma_t(i) = \Pr\left[s_t = i \middle| \bar{O}, \Gamma \right] = \frac{\alpha_t(i)\beta_t(i)}{\Pr\left[\bar{O} \middle| \Gamma \right]}; \quad i = 1,2,...,N \tag{15.68}$$

Mã chương trình Matlab cơ bản để thực hiện tính toán này như sau:

```
%Phiên bản 1:
gamma       = zeros(len,states);          % Phân bổ nhớ
for i=1:len
    for j=1:states
        gamma(i,j)  = alpha(i,j)*beta(i,j);% Tính toán biến gamma variable
    end
    gamma(i,:)=gamma(i,:)/sum(gamma(i,:));
end
```

Ta thấy rằng, không nhất thiết phải lưu giữ tất cả các giá trị này trong bộ nhớ, vì chúng có thể được lấy tổng đối với các chỉ số thời gian. Đoạn mã chương trình Matlab dưới đây giải thích rõ hơn về điều này.

```
%Phiên bản 2:
gamma_sum       = zeros(1,states);              % Phân bổ nhớ
for t = 1:len
    gamma_sum   = gamma_sum + alpha(t,:).*beta(t,:);
end
```

Đại lượng $\xi_t(i,j)$ được định nghĩa bởi

$$\xi_t(i,j) = \Pr\left[s_t = i, s_{t+1} = j \middle| \bar{O}, \Gamma \right] = \frac{\alpha_t(i).a_{ij}.b_j(O_{t+1}).\beta_{t+1}(j)}{\Pr\left[\bar{O} \middle| \Gamma \right]} \tag{15.69}$$

có thể được tính toán như sau:

```
%Phiên bản 1:
prob_model_given_seq    = zeros(1,len);       % Phân bổ nhớ cho mô hình
eta                     = zeros(states,states,len);% Phân bổ nhớ cho eta
for t=1:len                                    % Bắt đầu vòng lặp
    for i =1:states
        prob_model_given_seq    = zeros(1,len);% phân bổ bộ nhớ
        temp(i)                 = alpha(t,i)*beta(t,i);
    end
    prob_model_given_seq(t)     = sum(temp);  % Xác suất của mô hình
end

for i=1:states
    for j=1:states
        for t =1:(len-1)
            eta(i,j,t)=( (alpha(t,i)*p(i,j)...
                *b(out(t+1)+1),j)*beta(t+1,j)));
        end
    end
end
```

Cần phải đặc biệt lưu ý rằng, chương trình Matlab cho thấy rằng, ta không nhất thiết phải lưu lại biến số này cho tất cả các chỉ số thời gian. Đúng hơn ta có thể tính toán:

```
%Phiên bản 2:
eta             = zeros(states,states);             % Phân bổ bộ nhớ
sum_eta         = zeros(states,states);             % Phân bổ bộ nhớ

for t = 1:(len-1)
    for i=1:states
        eta(i,:) = ((alpha(t,i)*(p(i,:)...
            *(b(out(t+1)+1),:))).*beta(t+1,:)));
    end
    sum_eta     = sum_eta + eta;   % cộng với các giá trị của eta
end
```

Tại đây, ta xác định xác suất chuyển dịch trạng thái sang-trạng thái mới $\hat{a}_{ij}$ theo:

$$\hat{a}_{ij} = \frac{\textit{Số lần chuyển dịch được mong đợi từ trạng thái i đến trạng thái j}}{\textit{Số lần chuyển dịch được mong đợi từ trạng thái i}} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{15.70}$$

Mã chương trình Matlab cơ bản là:

```
%Phiên bản 1:
for i=1:states
    for j =1:states
```

```
        p_estimate(i,j)        =        sum(eta(i,j,:))/(sum(gamma(:,i))-
gamma(len,i));
    end
    p_estimate(i,:)= p_estimate(i,:)/sum(p_estimate(i,:));% chuẩn hóa
end
```

Hay đoạn mã chương trình:

```
%Phiên bản 2:
for i =1:sates
    for j =1:states
        p_estimate(i,j)     = sum_eta(i,j)...
            /(gamma(i)-alpha(len,i).*beta(len,i)...
            /(sum(alpha(len,:).*beta(len,:))));
    end
    p_estimate(i,:)     = p_estimate(i,:)/sum(p_estimate(i,:));
end
```

Tiếp theo $\hat{b}_j(e_k)$ được định nghĩa bởi:

$$\hat{b}_j(e_k) = \frac{\textit{Số lần mong đợi } e_k \textit{ được phát đi từ trạng thái } j}{\textit{Số lần thăm lại mong đợi đến trạng thái } j} = \frac{\sum_{t=1|O_t=e_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{15.71}$$

Được tính toán. Mã chương trình Matlab cơ bản là:

```
%Phiên bản 1:
for j =1:states
    i   = find(out==0);              % Tìm các chỉ số của các biến đúng
for k = 1:length(i)
        sum_gamma       = sum_gamma + gamma(i(k),j);
    end
    b(1,j)              = sum_gamma/sum(gamma(:,j));
    sum_gamma           = 0;
end
for j = 1:sates
    ii = find(out==1);             % Tìm các chỉ số của các lỗi
    for k = 1:length(ii)
        sum_gamma       = sum_gamma + gamma(ii(k),j);
    end
    b(2,j)              = sum_gamma/sum(gamma(:,j));
    sum_gamma           = 0;
end
for i = 1:states
    b(:,i)              = b(:,i)/sum(b(:,i));
end
```

Dùng các tính toán hiệu quả hơn, ma trận xác suất ký hiệu đầu ra có thể được ước tính là:

```
%Phiên bản 2:
out_0       = find(out==0);  % Tìm các chỉ số của các bit đúng
out_1       = find(out==1);  % Tìm các chỉ số của các lỗi
```

```
sum_0       = zeros(1,states);
sum_1       = zeros(1,states);
gamma_sum   = sum(gamma);

for i =1:length(out_0)
    sum_0   = sum_0 +gamma(out_0(i),:);   % Cộng với các bit đúng
end
for i=1:states
    for j =1:2
        if j==1
           b(j,i)  = sum_0(i)/gamma_sum(i); % Các phần tử b,các bit lỗi
        end
    end
end
for i = 1:states
    b(:,i)  = b(:,i)/sum(b(:,i));
end
```

Ta cũng có thể tính

$$\hat{\pi} = \left(\textit{số lần mong đợi ở trạng thái } S_i \textit{ tại thời điểm } t = 1\right) = \alpha_1(i).\beta_1(i) \tag{15.72}$$

Bước 3: Trở lại bước 1 với các giá trị mới của $\hat{\Gamma} = \left\{\hat{A}, \hat{B}, \hat{\pi}\right\}$, hay tương đương $\hat{\Gamma} = \Gamma$, đạt được bước 2 và lặp lại đến khi đạt được mức hội tụ mong muốn như được đề cập dưới đây.

15.5.1. Lấy tỉ lệ

Các véc tơ tiến và lùi có khuynh hướng tiến tới mũ 0 khi kích thước dữ liệu lớn và phải được lấy tỉ lệ phù hợp để ngăn ngừa thiếu hụt số hạng. Hằng số tỉ lệ, cách thức dùng nó được thực hiện trong mã chương trình Matlab được cho ở file **NVD15_bwa.m** trong phụ lục 15A, trước tiên được định nghĩa là:

$$C_t = \sum_{i=1}^{N} \alpha_t(i) \tag{15.73}$$

Các giá trị của $\alpha_t(j)$ được lấy tỉ lệ, được ký hiệu là $\overline{\alpha}_t(j)$ và được cho bởi:

$$\overline{\alpha}_t(j) = \frac{\alpha_t(j)}{C_t} \tag{15.74}$$

Tất nhiên, điều này có nghĩa là:

$$\sum_{i=1}^{N} \overline{\alpha}_t(i) = 1 \tag{15.75}$$

Các giá trị của C_t được lưu lại và được dùng để lấy tỉ lệ các biến lùi. Các giá trị được lấy tỉ lệ của $\beta_t(i)$, được ký hiệu là $\overline{\beta}_t$, được cho bởi:

$$\overline{\beta}_t(i) = \frac{\beta_i(i)}{C_t} \tag{15.76}$$

Với giá trị khởi tạo:

$$\bar{\beta}_T = \frac{[\mathbf{1}]}{C_T}$$

Trong đó [**1**] ký hiệu cho véc tơ cột chứa tất cả số 1. Biến γ cũng có thể được chuẩn hóa nếu muốn, nhưng việc lấy tỉ lệ các biến γ là không cần thiết.

15.5.2. Tiêu chuẩn ngừng và mức độ hội tụ

Vì thuật toán Baum-Welch là lặp, nên cần phải xác định số lần lặp cần thiết để đáp ứng mức độ chính xác của mô hình. Có lẽ cách tốt nhất để thực hiện điều này là hiển thị các ước tính của A và B khi thuật toán đang thực hiện. Nếu muốn mỗi phần tử của A và B chính xác đến một số con số ý nghĩa, thì cho phép liên tục thực hiện thuật toán cho tới khi các phần tử của A và B không còn thay đổi nữa, trong mức độ chính xác cho trước, từ lần lặp này tới lần lặp khác. Sau đó, thuật toán được kết thúc bằng nhân công. Kỹ thuật này khá hấp dẫn, vì biết được mức độ chính xác. Hơn nữa, dựa vào hiểu biết có thể đơn giản hóa một số lần lặp.

Phương pháp được dùng phổ biến khác để xác định tính hội tụ là liên tục lặp cho tới khi các giá trị kế tiếp của $\Pr[\bar{O} \mid \Gamma]$ lệch nhau rất nhỏ. (Thuật toán Baum-Welch đảm bảo hội tụ tới nghiệm khả năng giống nhất ML. Chứng minh điều này được thấy ở nhiều tài liệu). Giá trị của $\Pr[\bar{O} \mid \Gamma]$ được xác định theo hằng số tỉ lệ Ct trong (15.73). Cụ thể:

$$\Pr[\bar{O} \mid \Gamma] = \prod_{t=1}^{T} C_t \tag{15.77}$$

Khi *T* lớn, con số này sẽ rất nhỏ và thường được biểu diễn là:

$$\log_{10} \Pr[\bar{O} \mid \Gamma] = \sum_{t=1}^{T} \log_{10} C_t \tag{15.78}$$

Được xem là tỉ lệ log-ML (lấy log của hàm khả năng giống nhất) và được minh họa ở trường hợp 1 trong phần 15.6.

Cũng cần chỉ ra rằng, đối với tập dữ liệu cho trước, các ước tính của A và B là không duy nhất trừ khi đã định rõ các ước tính khởi tạo của A, B và Π. Vì véc tơ lỗi là một hàm của A và B, nên các kết hợp khác nhau có thể tạo ra các kết quả thống kê tương đương và kết quả cụ thể sẽ phụ thuộc vào các điều kiện khởi tạo (điều kiện đầu).

15.5.3. Mô hình Markov tương đương khối

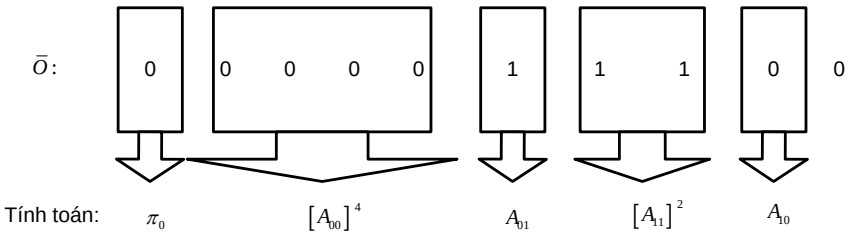
Thuật toán Baum-Welch là một trong nhiều thuật toán *ước tính lại* khả dụng để tính toán các tham số của mô hình dựa trên véc tơ lỗi. Chuỗi lỗi này có thể kéo theo hàng nghìn, hoặc thậm chí hàng triệu ký hiệu. Lưu ý rằng, để ước tính chính xác giá trị nhỏ của a_{ij} thì chu kỳ quan sát phải đủ lâu. Trong các hệ thống truyền thông, các giá trị nhỏ này thường là giá trị tới hạn để ước tính hiệu năng hệ thống. Độ phức tạp tính toán liên quan đến thuật toán Baum - Welch khá lớn khi véc tơ lỗi dài, vì các biến tiến và lùi được tính toán đối với mỗi ký hiệu

trong chuỗi lỗi đã cho. Hơn nữa, tính hội tụ chậm làm tăng thêm độ phức tạp tính toán. Với các ứng dụng xác suất lỗi thấp, véc tơ lỗi có chuỗi dài liên tiếp số 0. Ngoài ra, véc tơ lỗi phải chứa một số lượng đáng kể các sự kiện lỗi để véc tơ lỗi định nghĩa chính xác mô hình. Trong các tình huống này, thuật toán Baum-Welch là không hiệu quả và cần phải có một thuật toán nhanh hơn.

Phương pháp Turin: Một phương pháp được Turin đề xuất để giải quyết vấn đề này, bao gồm quá trình tính toán các biến tiến và biến lùi dùng phiên bản ma trận khối dạng:

$$A = \begin{bmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{bmatrix} \tag{15.79}$$

Quan hệ giữa các số hạng trong ma trận được định nghĩa bởi (15.79) được minh họa ở hình 15.11. Lưu ý rằng, số mũ của các ma trận con trong các tính toán có thể được tính toán trước (tiền tính toán-precompute) và được dùng *lại* để giảm toàn bộ tải tính toán.

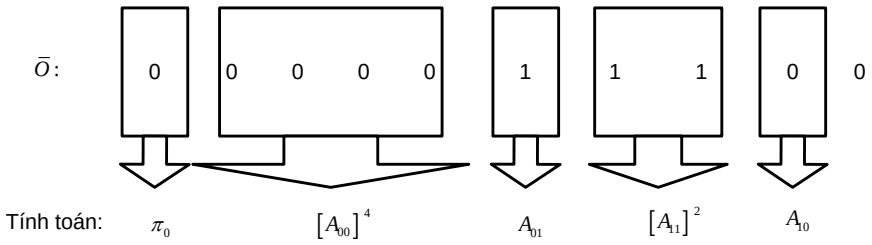


Hình 15.11: Các tính toán cho chuỗi 0000011100 (phiên bản 1)

Phương pháp Sivaprakasam và Shanmugan: Một thay đổi khác được Sivaprakasam và Shanmugan đề xuất dựa trên thực tế rằng đối với mô hình Markov tổng quát là có mô hình Fritcham thống kê tương đương có *k* trạng thái tốt và *N-k* trạng thái xấu và một ma trận A có dạng:

$$A = \begin{bmatrix} \Lambda_{00} & A_{01} \\ A_{10} & \Lambda_{11} \end{bmatrix} \tag{15.80}$$

Trong đó Λ_{00} *và* Λ_{11} là các ma trận đường chéo. Các mô hình này thường được coi là các mô hình Markov đường chéo khối. Với mô hình này, kênh giữ nguyên cùng trạng thái trong khoảng thời gian một cụm lỗi và chỉ thay đổi trạng thái ở thời điểm cuối một cụm. Kết quả là, tất cả các biến chỉ được tính toán tại các bước thời gian làm thay đổi ký hiệu lỗi. Nói cách khác, các biến được tính toán tại thời điểm bắt đầu của mỗi cụm các lỗi chứ không phải là trên mỗi ký hiệu. Bằng cách thay đổi này, các tính toán có dạng được minh họa ở hình 15.12. Tại đây, lưu ý rằng các tính toán trong khoảng thời gian các cụm dài liên tiếp số 1 và số 0 liên quan với việc làm tăng lũy thừa (số mũ) của ma trận đường chéo chứ không phải là làm tăng lũy thừa của các ma trận bất kỳ, và các phép nhân ma trận chỉ xảy ra tại thời điểm chuyển từ cụm này sang cụm khác. Vì vậy, rất hiệu quả tính toán, và các cụm lỗi dài có thể được xử lý mà không phải tính toán và lưu trữ quá mức.



Hình 15.12: Các tính toán cho chuỗi 0000011100 (phiên bản 2)

Sử dụng mô hình này ta chỉ cần quan tâm các độ dài liên tiếp các số 1 hoặc 0. Vì vậy, véc tơ lỗi có thể được đặt trong dạng khối rất kết đặc. Như một ví dụ minh họa đơn giản là, xét một chuỗi lỗi:

$$E = \left[\underbrace{00000}_{0^5} \underbrace{111}_{1^3} \underbrace{0000000000000}_{0^{14}} \underbrace{11}_{1^2} \underbrace{0000000000000000000000}_{0^{21}} \underbrace{100}_{0^2} \right] \tag{15.81}$$

Véc tơ độ dài liên tiếp tương ứng chuỗi lỗi này được viết là:

$$V = \left[0^5 1^3 0^{14} 1^2 0^{21} 1 0^2 \right] \tag{15.82}$$

Mã chương trình Matlab **NVD15_seglength.m** được cho ở phụ lục 15A, tạo ra véc tơ độ dài liên tiếp được định nghĩa bởi (15.82) từ véc tơ lỗi được định nghĩa bởi (15.81).

Việc triển khai các giải thuật ước tính ***lại*** dựa trên các mô hình Markov tương đương khối không được đề cập ở đây. Chương trình Matlab dựa trên công trình của Sivaprakasam và Shanmugan được cho ở file **NVD15_SemiMarkov.m** trong phụ lục 15A. Ví dụ minh họa cách dùng thuật toán này được trình bày ở phần sau

Không kể đến các thuật toán đã được dùng, mô hình Markov cho kênh rời rạc phải được tính toán cho các giá trị khác nhau của các tham số của kênh vật lý cơ bản. *Nếu kênh vật lý lớp dưới bị thay đổi theo bất kỳ cách nào, thì mô hình kênh rời rạc phải được ước tính lại*. Ví dụ, nếu một mô hình kênh rời rạc được triển khai cho một kênh đã cho, thì mô hình Markov phải được triển khai từ các mô phỏng mức dạng sóng hoặc các đo đạc cho các giá trị khác nhau của các tham số kênh bao gồm cả SNR. Vì vậy, tập các mô hình Markov được thông số hóa cho kênh lớp dưới có thể được triển khai và được sử dụng để thiết kế và phân tích các bộ mã hóa kiểm soát lỗi, các bộ đan xen, v.v…

15.6. Hai trường hợp điển hình

Ta kết thúc chương bằng hai trường hợp điển hình với mục đích làm sáng tỏ việc xác định kênh Markov và xác định mô hình bán Markov như sau:

Trường hợp 1, véc tơ lỗi được tạo ra trong ví dụ 15.3 được dùng làm đầu vào thuật toán Baum-Welch. Kết quả là một mô hình kênh được ước tính. Kết quả phân bố độ dài liên tục và xác suất lỗi từ mô hình này được xác định và sau đó được so sánh với phân bố độ dài liên tục và xác suất lỗi của chuỗi gốc được tạo ra.

Trường hợp 2, về cơ bản giống như trường hợp 1. Khác nhau chủ yếu là dùng mô hình *bán* Markov tương đương khối chứ không phải là mô hình Markov. Động cơ để dùng mô hình bán Markov là, mô hình bán Markov làm giảm đáng kể thời gian cần thiết để rút ra mô hình từ dữ liệu mô phỏng hoặc dữ liệu đo.

15.6.1. Trường hợp 1:

Trong trường hợp này, ta tạo ra một chuỗi lỗi có chiều dài L = 20.000 với một mô hình 2 trạng thái cho trước được định nghĩa bởi ma trận A và B sau:

$$A = \begin{bmatrix} 0,95 & 0,05 \\ 0,10 & 0,90 \end{bmatrix} \tag{15.83}$$

$$B = \begin{bmatrix} 0,95 & 0,50 \\ 0,05 & 0,50 \end{bmatrix} \tag{15.84}$$

Sau đó thuật toán Baum-Welch được dùng để ước tính mô hình sử dụng chuỗi lỗi đã được tạo ra. Lưu ý rằng ta, ta thừa nhận một mô hình ba trạng thái khi ứng dụng mô hình Baum-Welch. Điều này có thể dường như lạ thường nhưng, cho trước dữ liệu đo hoặc mô phỏng, thì mô hình kênh tạo ra dữ liệu không được biết. Hội thoại Matlab, với một số hội thoại không cần thiết được loại bỏ như sau (nghiên cứu kỹ chương trình Matlab):

```
>> NVD15_errvector
    Chấp nhận các giá trị mặc định ?
    Nhập y nếu đồng ý hoặc n nếu không:        n
    Nhập số các điểm được tạo ra L:            20000
    Nhập ma trận chuyển dịch trạng thái A:     [0,95 0,05; 0,1 0,9]
    Nhập ma trận phân bố lỗi B:                [0,95 0,5; 0,05 0,5]
>> Đầu ra 1 = đầu ra;
>> NVD15_bwa(20,3,đầu ra)
Nhập ma trận chuyển dịch trạng thái khởi tạo P:
    [0,9 0,05 0,05; 0,1 0,8 0,1; 0,1 0,2 0,7]
Nhập véc tơ xác suất trạng thái khởi tạo pye: [0,3 0,3 0,4]
Nhập ma trận xác suất ký hiệu đầu ra khởi tạo B: [0,9 0,8 0,7; 0,1 0,2 0,3]
```

Sau một lần lặp ta có (cần lưu ý rằng, chạy lại chương trình này có thể dẫn đến các ước tính cho A và B có thể khác với kết quả ở đây, thậm chí sử dụng cùng tham số các điều kiện đầu. Lý do là véc tơ lỗi dựa vào các ước tính của A và B là một hàm mẫu của quá trình ngẫu nhiên. Ngoài ra, các hàng của $\hat{A}$ *và các cột của* $\hat{B}$ có thể có tổng không phải là 1 khi mà các phần tử của $\hat{A}$ *và* $\hat{B}$ được biểu diễn với độ chính xác hạn chế).

$$\hat{A}_1 = \begin{bmatrix} 0,9051 & 0,0469 & 0,0481 \\ 0,0991 & 0,7931 & 0,1078 \\ 0,0895 & 0,1856 & 0,7249 \end{bmatrix} \qquad \hat{B}_1 = \begin{bmatrix} 0,9206 & 0,7462 & 0,5848 \\ 0,0794 & 0,2538 & 0,4152 \end{bmatrix}$$

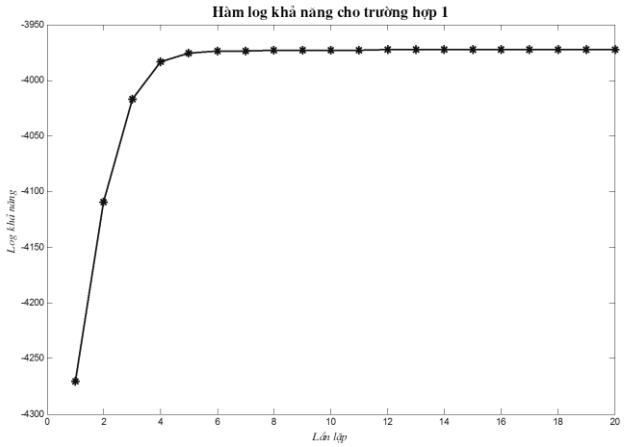
Sau 10 lần lặp ma trận chuyển dịch trạng thái được ước tính là:

$$\hat{A}_{10} = \begin{bmatrix} 0,9415 & 0,0300 & 0,0285 \\ 0,1157 & 0,7504 & 0,1340 \\ 0,0669 & 0,1382 & 0,7949 \end{bmatrix} \qquad \hat{B}_{10} = \begin{bmatrix} 0,9547 & 0,6745 & 0,4157 \\ 0,0453 & 0,3255 & 0,5743 \end{bmatrix}$$

Tại lần lặp thứ 20 là điểm kết thúc, ma trận chuyển dịch trạng thái được ước tính là:

$$\hat{A}_{20} = \begin{bmatrix} 0,9437 & 0,0299 & 0,0263 \\ 0,1173 & 0,7425 & 0,1402 \\ 0,0663 & 0,1324 & 0,8013 \end{bmatrix} \qquad \hat{B}_{20} = \begin{bmatrix} 0,9522 & 0,6637 & 0,4313 \\ 0,0478 & 0,3363 & 0,5687 \end{bmatrix}$$

Hàm log khả năng giống được minh họa ở hình 15.13. Cho thấy, hàm log khả năng hội tụ sau khoảng 10 lần lặp. Lưu ý rằng kết luận này phù hợp với các tính toán trước đây cho A_k và B_k khi $k = 10$ và $k = 20$. Cũng cần lưu ý rằng, như thảo luận trước đây, các con số khả năng giống là rất nhỏ.



Hình 15.13: Hàm log khả năng cho trường hợp 1

Tại đây, ta tạo chuỗi thứ hai *out2* dùng mô hình được ước tính bởi thuật toán Baum-Welch. Điều này cho ta với hội thoại khảo sát các giá trị mặc định được ẩn đi:

```
>> a = [0,9437 0,0299 0,0263; 0,1173 0,7425 0,1402; 0,0663 0,1324 0,8013]
>> b = [0,9522 0,6637 0,4313; 0,0478 0,3363 0,5687];
>> NVD15_errvector
    Chấp nhận các giá trị mặc định ?
    Nhập y cho yes hay n cho no > n
```

Nhập số các điểm được tạo ra L > 20000

Nhập ma trận chuyển dịch trạng thái A > a

Nhập ma trận phân bố lỗi B > b

>> Đầu ra 2 = đầu ra;

Để tính toán và vẽ $\Pr\{0^m|1\}$ cho cả out1 và out2, ta thực hiện chạy đoạn chương trình Matlab sau:

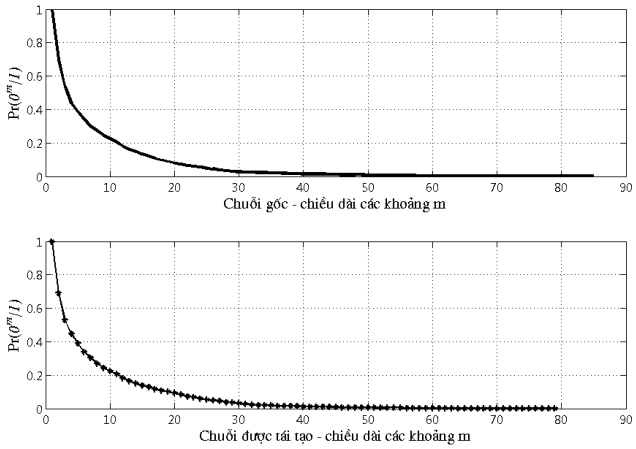
```
>> runcode1 = NVD15_seglength(out1);
>> runcode2 = NVD15_seglength(out2);
>> NVD15_interval (runcode1, runcode2);
```

Kết quả được minh họa ở hình 15.14. Trong đó nửa trên minh họa $\Pr\{0^m|1\}$ cho dữ liệu gốc và hình dưới minh họa $\Pr\{0^m|1\}$ dùng dữ liệu được tạo ra bởi mô hình được ước tính bởi giải thuật Baum-Welch.

Có một vài điểm khác nhau giữa hai hình được minh họa ở hình 15.14, đặc biệt trong lân cận m = 10. Tuy nhiên lưu ý rằng, xác suất lỗi được cho bởi.

>>p1 = sum(out1)/L kết quả là P1 = 0,0038

>>p2 = sum(out2)/L kết quả là P2 = 0,0035



Hình 15.14: $\Pr\{0^m|1\}$ *đối với dữ liệu gốc và dữ liệu từ mô hình được ước tính*

15.6.2. Trường hợp 2

Xét trường hợp kiểm tra được Sivaprakasam & Shammugan dùng để minh họa mô hình Markov đường chéo khối. Giả sử một mô hình ba trạng thái có hai trạng thái tối và một trạng thái xấu được định nghĩa bởi (lưu ý rằng ma trận chuyển dịch trạng thái A biểu diễn kênh vật lý và không có dạng đường chéo khối. Ma trận chuyển dịch trạng thái được ước tính của A là $\hat{A}$ sẽ có dạng đường chéo khối):

$$A = \begin{bmatrix} 0,90 & 0,02 & 0,08 \\ 0,10 & 0,50 & 0,40 \\ 0,10 & 0,20 & 0,70 \end{bmatrix} \tag{15.85}$$

Vì các trạng thái tốt là không lỗi và các lỗi luôn được tạo ra trong trạng thái xấu, nên ma trận quan sát lỗi được định nghĩa bởi:

$$B = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{15.86}$$

Lưu ý rằng, mô hình này là mô hình *bán ẩn*. Nếu xảy ra một lỗi, ta biết rằng lỗi đó được tạo ra bởi trạng thái 3. Tuy nhiên nếu lỗi không được tạo ra, thì không nhận biết được trạng thái đó.

Trước hết, ta tạo một véc tơ lỗi có độ dài 200.000, với ma trận chuyển dịch trạng thái A và ma trận quan sát lỗi B. Các câu lệnh Matlab, với ẩn hội thoại không cần thiết như sau.

```
>> a1 = [0,90 0,02 0,08; 0,10 0,50 0,40; 0,10 0,20 0,70];
>> b1 = [1 1 1; 0 0 1];
>> NVD15_errvector
Chấp nhận các giá trị mặc định?
    Nhập y cho yes hay n cho no > n
    Nhập số các điểm được tạo ra L > 20000
    Nhập ma trận chuyển dịch trạng thái A > a1
    Nhập ma trận phân bố lỗi B > b1
>> out1 = out;
>> runcode1 = NVD15_seglenght(out1)
>>[A_matrix, pi_est] = NVD15_semiMarkov(runcode1,100,[2 1]);
```

Lưu ý rằng, ta phải chạy chương trình **NVD15_seglenght** trước khi chạy chương trình **NVD15_semiMarkov** để tạo véc tơ độ dài liên tục.

Sau 50 lần lặp, ước tính ma trận chuyển dịch trạng thái là:

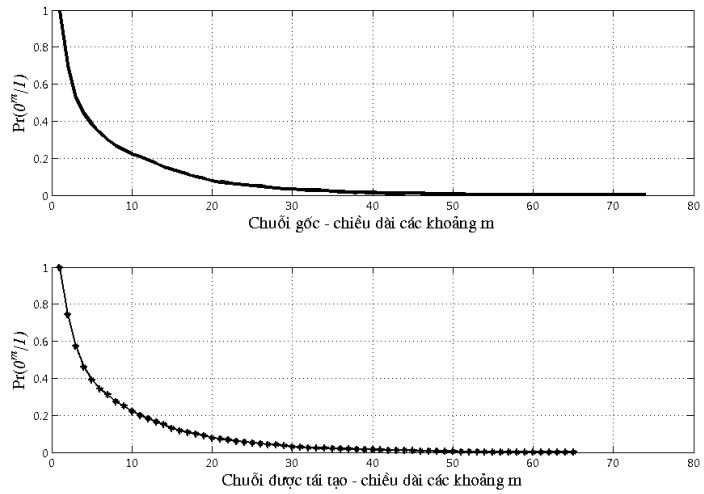
$$\hat{A} = \begin{bmatrix} 0,9043 & 0,0 & 0,0957 \\ 0,0 & 0,4911 & 0,5089 \\ 0,1402 & 0,1515 & 0,7083 \end{bmatrix} \tag{15.87}$$

Lưu ý rằng đây là gần sát với kết quả đạt được bởi Sivaprakasam & Shammugan . Có một số giải thích cho các khác nhau này. *Trước hết,* véc tơ lỗi được dựa trên mô hình được ước tính là một hàm mẫu của quá trình stochastic và với mục đích thực tế thì hai véc tơ lỗi là không đồng nhất. *Ngoài ra*, các véc tơ lỗi có độ dài khác nhau sẽ dẫn đến các môi hình hơi khác nhau. *Cuối cùng*, ta đã cố định số lần lặp và Sivaprakasam & Shammugan dùng tiêu chuẩn ngừng khác.

Bước tiếp theo là tạo véc tơ lỗi thứ hai dựa trên mô hình được ước tính và vẽ $\Pr\{0^m|l\}$ cho cả véc tơ lỗi gốc và véc tơ lỗi được tạo ra bởi mô hình được ước tính. Điều này được thực hiện dùng mã chương trình Matlab dưới đây.

```
>> a2 = [0.9043 0.0 0.0957 0.04911 0.45089; 0.1402 0.1515 0.7083];
>> b2 = [1 1 0;0 0 1];
>> NVD15_errvector
    Chấp nhận các giá trị mặc định ?
    Nhập y cho yes hay n cho no > n
    Nhập số các điểm được tạo ra L > 20000
    Nhập ma trận chuyển dịch trạng thái A > a2
    Nhập ma trận phân bố lỗi B > b2
>> out2 = out;
>> runcode2 = NVD15_seglenght(out2)
>> NVD15_intervals2 (runcode1, runcode2);
```

Các kết quả được minh họa ở hình 15.15. Lưu ý rằng, các kết quả là gần giống dù là mô hình gốc được định nghĩa bởi A và mô hình được ước tính được định nghĩa bởi $\hat{A}$ là hơi khác. Rõ rằng, ta biết từ bắt đầu bài toán này là A và $\hat{A}$ có thể sẽ khác vì $\hat{A}$ được giam giữ có dạng đường chéo khối.



Hình 15.15: $\Pr\{0^m|l\}$ *cho trường hợp 2*

Cũng là bài học để kiểm tra các xác suất lỗi. Để thực hiện điều này ta tính toán

```
>> pe1 = sum(out1)/20000; kết quả là Pe1 = 0.3617;
>> pe2 = sum(out2)/20000; kết quả là Pe2 = 0.3538;
```

Một khi thực hiện lại ta có giá trị gần giống.