

Song song hóa thời gian suy diễn trong mô hình Markov ẩn

Temporal Parallelization of Inference in Hidden Markov Models

Lê Nhựt Nam (18120061) Trần Đại Chí (18127070)

Trần Xuân Lộc (18127131)

Khoa Công nghệ Thông tin, Trường Đại học Khoa học Tự nhiên, Đại học Quốc gia TP.HCM

Ngày 12 tháng 8 năm 2022

Nội dung

- 1 Giới thiệu
- 2 Các công trình liên quan
- 3 Kiến thức nền tảng
- 4 Các phương pháp đề xuất
- 5 Kết quả thực nghiệm
- 6 Kết luận
- 7 Tài liệu tham khảo

Giới thiệu bài toán

Mô hình markov ẩn (HMM) là gì?

Là mô hình thống kê đặc trưng bởi tính đơn giản và khắc phục được những điểm yếu của mô hình Markov khi đối tượng quan sát có nhiều trạng thái và xác suất chuyển đổi trạng thái thay đổi theo thời gian.

Phương pháp song song (Parallelization) là gì?

Nhằm mục đích giảm chi phí tính toán và có thể mở rộng cho các bài toán trong thực tế. Sử dụng một số phần cứng chuyên dụng như GPUs, NPU, TPUs.

Bài toán suy diễn (inference problem) là gì?

Trong bối cảnh của HMM, bài toán suy diễn dựa trên dữ liệu từ quan sát được để suy luận ra xác suất xảy ra các sự kiện trong tương lai thông qua một phương pháp phân tích xác suất nào đó.

Động lực nghiên cứu

Các công trình nghiên cứu hiện tại tập trung vào việc cải thiện tốc độ thuật toán viterbi thông qua phương pháp song song hóa (parallelization) nhưng chưa được nghiên cứu kỹ lưỡng mà chỉ tập trung tối ưu mục tiêu (nhận dạng tiếng nói).

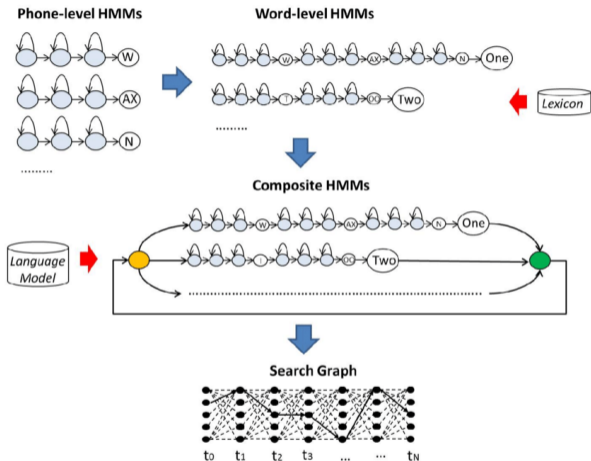
Hiện tại phương pháp song song hoá sử dụng thuật toán parallel-scan đã được ứng dụng trên thuật toán Bayesian filtering and smoothing để tăng tốc độ tính toán tuần tự. Trong bài báo này, tác giả sẽ tận dụng phương pháp này và áp dụng nó trên mô hình HMM cho bài toán suy luận (inference problem) với cơ chế tính toán khác.

Ý nghĩa thực tiễn

Một số ứng dụng của mô hình Markov ẩn trong thực tế như:

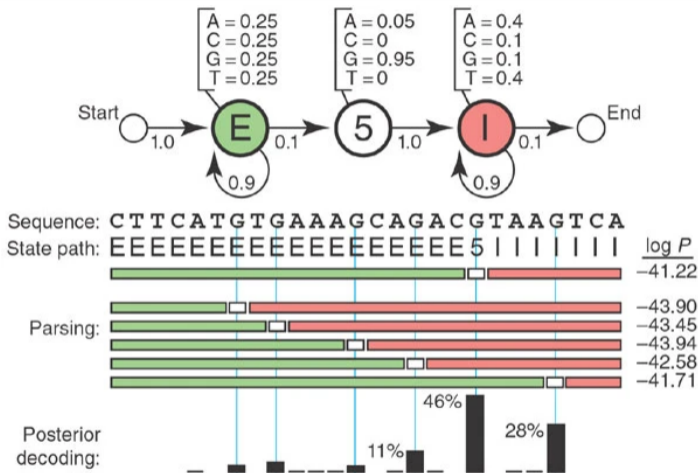
- Nhận dạng: tiếng nói, cử chỉ, nét mặt.
- Theo dõi mục tiêu.
- Lĩnh vực tin sinh học: dự đoán gen.
- Giải mã mã morse.

Một số ví dụ về ứng dụng trong thực tế I



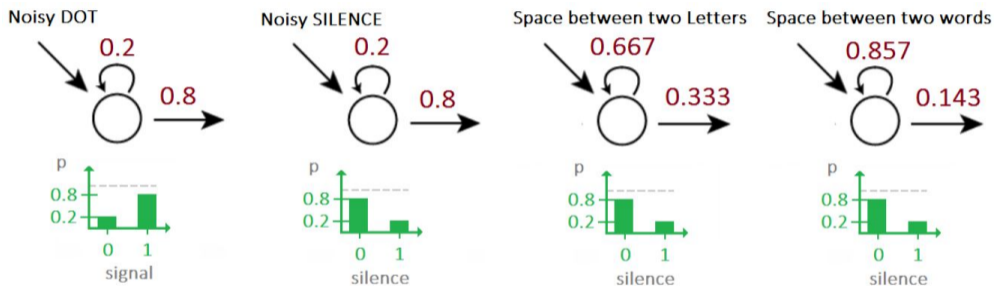
Hình 1: Ứng dụng trong nhận dạng tiếng nói [12].

Một số ví dụ về ứng dụng trong thực tế II



Hình 2: Ứng dụng trong dự đoán trình tự gen [5].

Một số ví dụ về ứng dụng trong thực tế III



Hình 3: Ứng dụng trong giải mã mã morse [1].

Các công trình liên quan I

Các phương pháp liên quan tăng tốc quá trình suy diễn của HMM:

| <i>Tác giả</i> | <i>Phương pháp</i> | <i>Ưu điểm</i> | <i>Nhược điểm</i> |
|--------------------|--|---|--|
| Lifshits et al.[8] | Sử dụng một lược đồ nén nhờ tận dụng các mẫu lặp đi lặp lại trong các chuỗi quan sát được. | Nhanh hơn Viterbi, có khả năng song song hóa cao. | Chỉ áp dụng tăng tốc cho giải mã và huấn luyện HMM. |
| Sand et al.[13] | Đề xuất một thư viện C++ sử dụng single instruction trên nhiều bộ xử lý dữ liệu và nhiều cores để tăng tốc Viterbi, forward, backward và Baum Welch. | Tăng tốc các thuật toán, giảm thời gian thực thi. | Thời gian thực thi ở pha giải mã hậu nghiệm vẫn còn chậm, chưa giải quyết vấn đề thừa trong các mô hình HMM. |

Bảng 1: Tổng hợp các công trình liên quan I.

Các công trình liên quan II

| <i>Tác giả</i> | <i>Phương pháp</i> | <i>Ưu điểm</i> | <i>Nhược điểm</i> |
|---------------------------|--|--|--|
| Nielsen and Sand[11] | Sử dụng parallel reduction để tăng tốc trong thuật toán forward. | Cần ít tương tác giữa các bộ xử lý và thực thi nhanh với số lượng trạng thái nhỏ. | Chậm với số lượng trạng thái lớn. |
| Chatterjee and Russell[2] | Sử dụng khái niệm temporal abstraction từ quy hoạch động để tăng tốc thuật toán Viterbi. | Đảm bảo sẽ tìm được Viterbi path với tốc độ thực thi nhanh hơn Viterbi và nhanh gấp đôi so với CFDP. | Giới hạn của TAV xuất hiện khi hệ thống có các chuyển đổi trạng thái thường xuyên. |

Bảng 2: Tổng hợp các công trình liên quan II.

Các công trình liên quan III

| <i>Tác giả</i> | <i>Phương pháp</i> | <i>Ưu điểm</i> | <i>Nhược điểm</i> |
|-------------------|---|--|---|
| Zhihui et al.[3] | Đề xuất tile-based Viterbi algorithm bằng cách sử dụng accelerated hardware, trong đó phép nhân ma trận được thực thi một cách song song. | Cho thấy hiệu suất vượt trội hơn các phương pháp wave-front và streaming, nhanh hơn Viterbi. | Chưa khai thác song song hóa trong tác vụ suy diễn HMM. |
| Maleki et al.[10] | Đề xuất phương pháp tối ưu mà cho phép giải một lớp bài toán quy hoạch động cụ thể bằng tropical semirings và dùng nó để tối ưu thuật toán Viterbi. | Tăng tốc đáng kể parallel Viterbi decoder lên 24x với 64 processors. | Chưa khai thác song song hóa trong tác vụ suy diễn HMM. |

Bảng 3: Tổng hợp các công trình liên quan III.

Các công trình liên quan IV

Các thuật toán được sử dụng cho nhiệm vụ suy luận HMM trên GPUs:

| <i>Thuật toán</i> | <i>Trạng thái</i> | <i>Quan sát</i> | <i>Tốc độ cải thiện</i> |
|----------------------|-------------------|-----------------|-------------------------|
| Forward-backward [7] | 8 | 200 | 3.5x |
| Forward [9] | 512 | 3-10 | 880x |
| Baum-Welch [9] | 512 | 3-10 | 180x |
| Viterbi [16] | — | 2000-3000 | 3x |
| Forward [6] | 4000 | 1000 | 6x |
| Baum-Welch [6] | 4000 | 1000 | 65x |

Bảng 4: Tổng hợp các công trình liên quan IV.

Các công trình liên quan V

Các công trình liên quan sử dụng parallel-scan để tăng tốc quá trình tính toán:

| <i>Tác giả</i> | <i>Phương pháp</i> | <i>Ưu điểm</i> |
|------------------------|--|--|
| S. Särkkä et al.[14] | Đề xuất phương pháp dựa trên IEKS+sigma-point bằng cách sử dụng một thuật toán duyệt để song song hóa. | Giảm độ phức tạp bao tuyến tính của các phương pháp ước lượng trạng thái về logarithm. |
| F. Yaghoobi et al.[15] | Định nghĩa các phần tử, toán tử cho tất cả các tổng tiền tố và thuật toán duyệt song song, song song hóa các phương trình Bayesian filtering và smoothing tổng quát. | Giảm độ phức tạp tuyến tính của các thuật toán smoothing tiêu chuẩn về logarithm. |

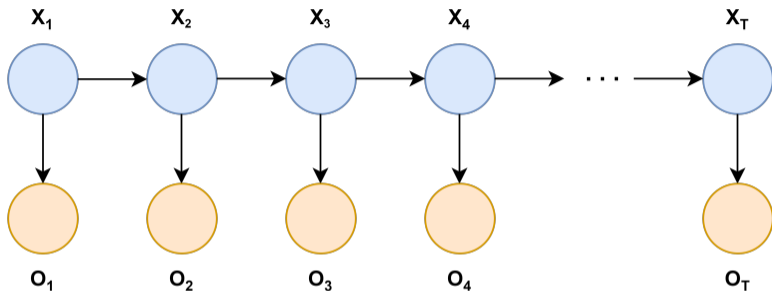
Bảng 5: Tổng hợp các công trình liên quan V.

Nhược điểm: lượng tính toán vẫn còn lớn so với các thuật toán tuần tự, cần phát triển particle filter and smoother methods

Ký hiệu

- D là số lượng các trạng thái (states).
- T là số lượng các quan sát (observations).
- $O = p(y_k|x_k)$ là ma trận quan sát (observation matrix).
- $x = (x_1, \dots, x_D)$ là tập các trạng thái.
- $y = (y_1, \dots, y_T)$ là tập các quan sát
- $\psi = (\psi_1, \dots, \psi_T)$ là tập các hàm tiềm năng (potential) dùng để xác định các clique của đồ thị.
- $\Pi = p(x_k|x_{k-1})$ là ma trận chuyển trạng thái (transition matrix).
- $I = (I_1, \dots, I_D)$ là tập các xác suất trước (prior probabilities) cho các trạng thái.

Cấu trúc HMM



Hình 4: Cấu trúc mô hình Markov ẩn.

- Dãy chuyển các trạng thái:

$$X = (X_1, \dots, X_T)$$

- Dãy các sự kiện quan sát được:

$$O = (O_1, \dots, O_T)$$

Định nghĩa bài toán

Giả sử mỗi biến x_D nhận một giá trị xác suất tương ứng cho mỗi trạng thái trong tập $\mathcal{X} = \{1, \dots, D\}$ và mỗi hàm tiềm năng sẽ là một hàm của một tập con của x_D mà được ký hiệu là x_{α_D} với $\alpha_D = (y_1, \dots, y_T)$ là các chỉ số. Phân phối kết hợp của các biến ngẫu nhiên được biểu diễn theo phân phối Gibbs như sau:

$$p(x) = \frac{1}{Z} \prod_{t=1}^T \psi_t(x_{\alpha_D}), \quad (1)$$

Trong đó $Z = \sum_x \prod_{t=1}^T \psi_t(x_{\alpha_D})$ là một hằng số chuẩn hóa.

Kiến thức nền tảng

Thông thường, một bài toán suy diễn là sự tính toán của tất cả các biên.

$$p(x_k) = \frac{1}{Z} \sum_{x \setminus x_k} \prod_{t=1}^T \psi_t(x_{\alpha_t}), \quad (2)$$

Trong đó $x \setminus x_k = (x_1, \dots, x_{k-1}, x_{k+1}, \dots, x_T)$ là tổng của tất cả các biên ngoại trừ x_k .

Ngoài ra, một nhiệm vụ suy diễn khác là bài toán tối đại số lượng tương đồng.

$$p^*(x_k) = \max_{x \setminus x_k} \prod_{t=1}^T \psi_t(x_{\alpha_t}), \quad (3)$$

Trong đó tối đa được tính toán đến tất cả các biên ngoại trừ x_k .

Kiến thức nền tảng

Trong mô hình Markov ẩn, bài toán suy diễn được đề cập như sau:

$$x_k \sim p(x_k|x_{k-1}), \quad (4a)$$

$$y_k \sim p(y_k|x_k), \quad (4b)$$

Trong đó \sim ký hiệu cho sự phân phối, chuỗi x_1, \dots, x_T là Markovian với các xác suất chuyển trạng thái $p(x_k|x_{k-1})$, $p(y_k|x_k)$ là các xác suất quan sát được và $x_1 \sim p(x_1)$ là một xác suất tiên nghiệm.

Trong Eq. 1, bài toán suy diễn được diễn giải như sau:

$$\psi_1(x_1) = p(y_1|x_1)p(x_1), \quad \text{với } k = 1. \quad (5a)$$

$$\psi_k(x_{k-1}, x_k) = p(y_k|x_k)p(x_k|x_{k-1}), \quad \text{với } k > 1. \quad (5b)$$

Kiến thức nền tảng

Từ (5a) và (5b), phân phối kết hợp trong Eq. 1 sẽ có dạng như sau:

$$p(x) = \frac{1}{Z} \psi_1(x_1) \prod_{t=2}^T \psi_t(x_{t-1}, x_t), \quad (6)$$

Trong đó $\alpha_k = (k - 1, k)$ nếu $k > 1$ và $\alpha_1 = (1)$ nếu $k = 1$.

Chúng ta có thể áp dụng các loại toán tử chung đến Eq. 6 như sau:

$$F(x_k) = \frac{1}{Z} OP_k \left(\psi_1(x_1) \prod_{t=2}^T \psi_t(x_{t-1}, x_t) \right), k = 1, \dots, T, \quad (7)$$

Trong đó $OP_k(\cdot)$ là một chuỗi các toán tử áp dụng đến tất cả các biến ngoại trừ x_k . Để tính được Eq. 7, chúng ta có thể sử dụng hai thuật toán là sum-product hoặc max-product cho các toán tử cộng và lấy cực đại. Tuy nhiên, vì hai toán tử này đã được tác giả chứng minh là có tính kết hợp nên ở những phần sau chúng sẽ được sử dụng với thuật toán duyệt song song để song song hóa sự tính toán cho hai thuật toán trên.

Thuật toán tổng tích cổ điển

Hàm tiềm năng thẳng được định nghĩa như sau:

$$\psi_{1,k}^f(x_k) = \sum_{x_{1:k-1}} \psi_1(x_1) \prod_{t=2}^k \psi_{t-1,t}(x_{t-1}, x_t), \quad (8)$$

Hàm tiềm năng ngược được định nghĩa như sau:

$$\psi_{k,T}^b(x_k) = \sum_{x_{k+1:T}} \prod_{t=k}^T \psi_{t,t+1}(x_t, x_{t+1}), \quad (9)$$

Trong đó $\psi_{T,T+1}(x_T, x_{T+1}) = 1$.

Chúng ta bây giờ có thể tính được phân phối biên $p(x_k)$ tương ứng đến toán tử cộng trong Eq. 7 như sau:

$$p(x_k) = \frac{1}{Z_k} \psi_{1,k}^f(x_k) \psi_{k,T}^b(x_k), \quad (10)$$

Trong đó $Z_k = \sum_{x_k} \psi_{1,k}^f(x_k) \psi_{k,T}^b(x_k)$.

Mã giả thuật toán tổng tích cổ điển

Input : Các hàm tiềm năng $\psi_k(\cdot)$ với $k = 1, \dots, T$.

Output: Hàm tiềm năng thẳng $\psi_{1,K}^f(x_k)$ và ngược $\psi_{k,T}^b(x_k)$ với $k = 1, \dots, T$.

//Forward pass:

$$\psi_{1,1}^f(x_1) = \psi_1(x_1)$$

for $k \leftarrow 2$ **to** T **do**

$$\left| \psi_{1,k}^f(x_k) = \sum_{x_{k-1}} \psi_{1,k-1}^f(x_{k-1}) \psi_{k-1,k}(x_{k-1}, x_k) \right.$$

end

//Backward pass:

$$\psi_{T,T}^b(x_T) = 1$$

for $k \leftarrow 2$ **to** $T - 1$ **do**

$$\left| \psi_{k,T}^b(x_k) = \sum_{x_{k+1}} \psi_{k,k+1}(x_k, x_{k+1}) \psi_{k+1,T}^b(x_{k+1}) \right.$$

end

▷ Tối ưu hóa

▷ Chạy tuần tự

▷ Tối ưu hóa

▷ Chạy tuần tự

Algorithm 1: Thuật toán tổng-tích cổ điển.

Ví dụ thuật toán tổng tích cổ điển

Giả sử chúng ta có tập các trạng thái $S = (\text{'Rainy': } 0, \text{'Sunny': } 1)$, chuỗi các quan sát $O = (\text{'walk': } 0, \text{'shop': } 1, \text{'clean': } 2)$, phân phối ban đầu $\pi = \{\text{'Rainy': } 0.6, \text{'Sunny': } 0.4\}$ và hai ma trận cho chuyển trạng thái và quan sát được như bên dưới

| | Rainy | Sunny |
|-------|-------|-------|
| Rainy | 0.7 | 0.3 |
| Sunny | 0.4 | 0.6 |

Bảng 6: Ma trận chuyển trạng thái

| | walk | shop | clean |
|-------|------|------|-------|
| Rainy | 0.1 | 0.4 | 0.5 |
| Sunny | 0.6 | 0.3 | 0.1 |

Bảng 7: Ma trận quan sát được

Nhiệm vụ của chúng ta là cần đi tìm hàm tiềm năng thẳng $\psi_{1,k}^f(x_k)$, hàm tiềm năng ngược $\psi_{k,T}^b(x_k)$ và phân phối biên $p(x_k)$ dựa vào các dữ kiện trên.

Ví dụ thuật toán tổng tích cổ điển

Ở bước forward, ban đầu $k = 1$, ta có

$$\psi_{1,1}^f(x_1) = \psi_1(x_1) = p(y_1|x_1)p(x_1) = \begin{bmatrix} 0.1 & 0.6 \end{bmatrix} \times \begin{bmatrix} 0.6 & 0.4 \end{bmatrix} = \begin{bmatrix} 0.06 & 0.24 \end{bmatrix}$$

Khi $k = 2$ đến $T = 3$

$$\psi_{1,k}^f(x_k) = \psi_{1,k-1}^f \cdot \psi_k(x_{k-1}, x_k) = \psi_{1,k-1}^f \cdot p(y_k|x_k)p(x_k|x_{k-1})$$

$$\rightarrow \psi_{1,2}^f(x_2) = \psi_{1,1}^f \cdot \psi_2(x_1, x_2) = \begin{bmatrix} 0.06 & 0.24 \end{bmatrix} \cdot \left(\begin{bmatrix} 0.4 & 0.3 \end{bmatrix} \times \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix} \right) = \begin{bmatrix} 0.0552 & 0.0486 \end{bmatrix}$$

$$\rightarrow \psi_{1,3}^f(x_3) = \psi_{1,2}^f \cdot \psi_3(x_2, x_3) =$$

$$\begin{bmatrix} 0.0552 & 0.0486 \end{bmatrix} \cdot \left(\begin{bmatrix} 0.5 & 0.1 \end{bmatrix} \times \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix} \right) = \begin{bmatrix} 0.02904 & 0.004572 \end{bmatrix}$$

Cuối cùng, ta có được hàm tiềm năng thẳng $\psi_{1,k}^f(x_k) = \begin{bmatrix} 0.06 & 0.24 \\ 0.0552 & 0.0486 \\ 0.02904 & 0.004572 \end{bmatrix}$

Ví dụ thuật toán tổng tích cổ điển

Ở bước backward, ban đầu $k = 3$, ta sẽ tối ưu hóa $\psi_{3,3}^b(x_3) = \begin{bmatrix} 1.0 & 1.0 \end{bmatrix}$

Khi $k = T - 1 = 2$ đến 1

$$\psi_{k,T}^b(x_k) = \psi_{k,k+1}(x_k, x_{k+1}) \cdot \psi_{k+1,T}^b(x_{k+1})$$

$$\rightarrow \psi_{2,3}^b(x_2) = \psi_{2,3}(x_2, x_3) \cdot \psi_{3,3}^b(x_3) = \left(\begin{bmatrix} 0.5 & 0.1 \end{bmatrix} * \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix} \right) \cdot \begin{bmatrix} 1.0 & 1.0 \end{bmatrix} =$$

$$\begin{bmatrix} 0.38 & 0.26 \end{bmatrix}$$

$$\rightarrow \psi_{1,3}^b(x_1) = \psi_{1,2}(x_1, x_2) \cdot \psi_{2,3}^b(x_2) =$$

$$\left(\begin{bmatrix} 0.4 & 0.3 \end{bmatrix} \times \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix} \right) \cdot \begin{bmatrix} 0.38 & 0.26 \end{bmatrix} = \begin{bmatrix} 0.1298 & 0.1076 \end{bmatrix}$$

$$\text{Cuối cùng, ta có được hàm tiềm năng thẳng } \psi_{k,T}^b(x_k) = \begin{bmatrix} 0.1298 & 0.1076 \\ 0.38 & 0.26 \\ 1.0 & 1.0 \end{bmatrix}$$

Ví dụ thuật toán tổng tích cổ điển

Sau khi đã có được hàm tiềm năng thẳng và hàm tiềm năng ngược, chúng ta sẽ tìm phân phối biên $p(x_k)$ bằng cách áp dụng Eq. 10

$$\begin{aligned}
 p(x_k) &= \frac{1}{Z_k} \psi_{1,k}^f(x_k) \psi_{k,T}^b(x_k) = \frac{\begin{bmatrix} 0.06 & 0.24 \end{bmatrix} \times \begin{bmatrix} 0.1298 & 0.1076 \end{bmatrix}}{\sum \begin{bmatrix} 0.06 & 0.24 \end{bmatrix} \times \begin{bmatrix} 0.1298 & 0.1076 \end{bmatrix}} + \\
 &\frac{\begin{bmatrix} 0.0552 & 0.0486 \end{bmatrix} \times \begin{bmatrix} 0.38 & 0.26 \end{bmatrix}}{\sum \begin{bmatrix} 0.0552 & 0.0486 \end{bmatrix} \times \begin{bmatrix} 0.38 & 0.26 \end{bmatrix}} + \frac{\begin{bmatrix} 0.02904 & 0.004572 \end{bmatrix} \times \begin{bmatrix} 1.0 & 1.0 \end{bmatrix}}{\sum \begin{bmatrix} 0.02904 & 0.004572 \end{bmatrix} \times \begin{bmatrix} 1.0 & 1.0 \end{bmatrix}} \\
 &= \begin{bmatrix} 0.2317 & 0.7683 \\ 0.6241 & 0.3759 \\ 0.8640 & 0.1360 \end{bmatrix}
 \end{aligned}$$

Vậy với chuỗi quan sát $O = \{0, 1, 2\}$ thì ta sẽ có được chuỗi trạng thái dự đoán được là $\{\text{'Sunny'}, \text{'Rainy'}, \text{'Rainy'}\}$.

Thuật toán duyệt song song

Thuật toán duyệt song song (parallel-scan) hay còn được gọi là tổng tiền tố (prefix sum) dùng để tính toán song song tất cả các tổng tiền tố của các toán tử kết hợp. Đưa ra một chuỗi các phần tử (a_1, \dots, a_T) và một toán tử kết hợp hai ngôi \otimes , tổng tiền tố của chuỗi này được tính như sau:

$$(a_1, a_1 \otimes a_2, \dots, a_1 \otimes a_2 \otimes \dots \otimes a_T). \quad (11)$$

Một cách tương tự, chúng ta cũng sẽ tính được tổng tiền tố cho chuỗi trên theo dạng đảo ngược như sau:

$$(a_1 \otimes a_2 \otimes \dots \otimes a_T, \dots, a_{T-1} \otimes a_T, a_T). \quad (12)$$

Mã giả thuật toán duyệt tuần tự

Input : Các phần tử a_k với $k = 1, \dots, T$ và toán tử \otimes .

Output: Tổng tiền tố a_k với $k = 1, \dots, T$.

for $k \leftarrow 0$ **to** $T - 1$ **do**

▷ Chạy tuần tự

| $a[i + 1] = a[i] \otimes a[i + 1]$

end

Algorithm 2: Thuật toán duyệt tuần tự.

Giả sử chúng ta có chuỗi $a = \{a_1, a_2, \dots, a_T\}$ và một toán tử kết hợp hai ngôi \otimes thì nếu sử dụng thuật toán 2, chúng ta có thể tính tổng tiền tố của chuỗi a với độ phức tạp $O(T)$. Tuy nhiên, chúng ta có thể tính tổng tiền tố của chuỗi này thông qua thuật toán duyệt song song với độ phức tạp chỉ còn $O(\log T)$.

Mã giả thuật toán duyệt song song I

Input : Các phần tử a_k với $k = 1, \dots, T$ và toán tử \otimes .

Output: Tổng tiền tố a_k với $k = 1, \dots, T$.

for $i \leftarrow 1$ **to** T **do**

| $b[i] \leftarrow a[i]$

end

//Up sweep:

for $d \leftarrow 0$ **to** $\log_2 T - 1$ **do**

| **for** $i \leftarrow 0$ **to** $T - 1$ **by** 2^{d+1} **do**

| | $a[i + 2^{d+1} - 1] \leftarrow a[i + 2^d - 1] \otimes a[i + 2^{d+1} - 1]$

| **end**

end

$a_T \leftarrow 0$ $\triangleright 0$ là phần tử trung lập cho toán tử \otimes

\triangleright Chạy song song

\triangleright Chạy song song

Mã giả thuật toán duyệt song song II

```
//Down sweep:
```

```
for  $d \leftarrow \log_2 T - 1$  to 0 do
```

```
  for  $i \leftarrow 0$  to  $T - 1$  by  $2^{d+1}$  do
```

▷ Chạy song song

```
    temp = a[i +  $2^d - 1$ ]
```

```
    a[i +  $2^d - 1$ ] = a[i +  $2^{d+1} - 1$ ]
```

```
    a[i +  $2^{d+1} - 1$ ] = temp  $\otimes$  a[i +  $2^{d+1} - 1$ ]
```

```
  end
```

```
end
```

```
//Final pass:
```

```
for  $i \leftarrow 1$  to  $T$  do
```

▷ Chạy song song

```
  | a[i]  $\leftarrow$  a[i]  $\otimes$  b[i]
```

```
end
```

Algorithm 3: Thuật toán duyệt song song.

Ví dụ thuật toán duyệt song song

Giả sử có một danh sách $a = (5, 3, -6, 2, 7, 10, -2, 8)$ và toán tử $+$, chúng ta sẽ sử dụng thuật toán tìm kiếm song song để tìm tổng tiền tố cho các phần tử a_k trong danh sách này. Ở bước Up sweep, vì $T = 8$ nên $d = 0, 1, 2$.

Với $d = 0$, bước nhảy của 2^{d+1} là 2.

- $i = 0, a[0 + 2^{0+1} - 1] += a[0 + 2^0 - 1] \rightarrow a[1] += a[0] = 3 + 5 = 8$
- $i = 2, a[2 + 2^{0+1} - 1] += a[2 + 2^0 - 1] \rightarrow a[3] += a[2] = 2 - 6 = -4$
- $i = 4, a[4 + 2^{0+1} - 1] += a[4 + 2^0 - 1] \rightarrow a[5] += a[4] = 10 + 7 = 17$
- $i = 6, a[6 + 2^{0+1} - 1] += a[6 + 2^0 - 1] \rightarrow a[7] += a[6] = 8 - 2 = 6$

Lúc này, a trở thành $(5, 8, -6, -4, 7, 17, -2, 6)$.

Với $d = 1$, bước nhảy của 2^{d+1} là 4.

- $i = 0, a[0 + 2^{1+1} - 1] += a[0 + 2^1 - 1] \rightarrow a[3] += a[1] = -4 + 8 = 4$
- $i = 4, a[4 + 2^{1+1} - 1] += a[4 + 2^1 - 1] \rightarrow a[7] += a[5] = 6 + 17 = 23$

Lúc này, a trở thành $(5, 8, -6, 4, 7, 17, -2, 23)$.

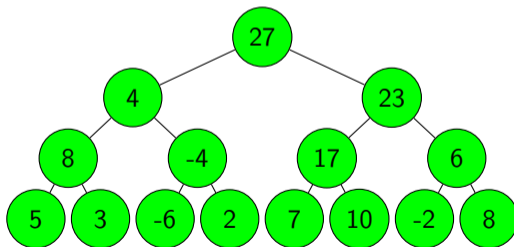
Ví dụ thuật toán duyệt song song

Với $d = 2$, bước nhảy của 2^{d+1} là 8

- $i = 0$, $a[0 + 2^{2+1} - 1] += a[0 + 2^2 - 1] \rightarrow a[7] += a[3] = 23 + 4 = 27$

Lúc này, a trở thành $(5, 8, -6, 4, 7, 17, -2, 27)$.

Ở trường hợp Up sweep, chúng ta có thể xem dãy a như là các nút lá của một cây nhị phân hoàn hảo và các nút bên trong sẽ được tính bởi tổng của tất cả các nút lá trong cây con của nó theo bước nhảy là 2^{d+1} với $d = 0, 1, 2$ theo thứ tự duyệt từ dưới đi lên ($\text{sum}[u] = \text{sum}[\text{Left}[u]] + \text{sum}[\text{Right}[u]]$ với u là nút).



Hình 5: Up sweep pass

Ví dụ thuật toán duyệt song song

Ở bước Down sweep, phần tử cuối cùng trong danh sách a vừa được cập nhật ở bước Up sweep sẽ trở thành 0 và ta cũng sẽ xét 3 trường hợp cho $d = 2, 1, 0$.

Với $d = 2$, bước nhảy của 2^{d+1} là 8

- $i = 0$, $\text{temp} = a[0 + 2^2 - 1] = a[3] = 4$
 $a[0 + 2^2 - 1] = a[0 + 2^{2+1} - 1] \rightarrow a[3] = a[7] = 0$
 $a[0 + 2^{2+1} - 1] += \text{temp} \rightarrow a[7] += \text{temp} \rightarrow a[7] = 4$

Lúc này, a trở thành $(5, 8, -6, 0, 7, 17, -2, 4)$.

Với $d = 1$, bước nhảy của 2^{d+1} là 4

- $i = 0$, $\text{temp} = a[0 + 2^1 - 1] = a[1] = 8$
 $a[0 + 2^1 - 1] = a[0 + 2^{1+1} - 1] \rightarrow a[1] = a[3] = 0$
 $a[0 + 2^{1+1} - 1] += \text{temp} \rightarrow a[3] += \text{temp} \rightarrow a[3] = 8$
- $i = 4$, $\text{temp} = a[4 + 2^1 - 1] = a[5] = 17$
 $a[4 + 2^1 - 1] = a[4 + 2^{1+1} - 1] \rightarrow a[5] = a[7] = 4$
 $a[4 + 2^{1+1} - 1] += \text{temp} \rightarrow a[7] += \text{temp} \rightarrow a[7] = 21$

Lúc này, a trở thành $(5, 0, -6, 8, 7, 4, -2, 21)$.

Ví dụ thuật toán duyệt song song

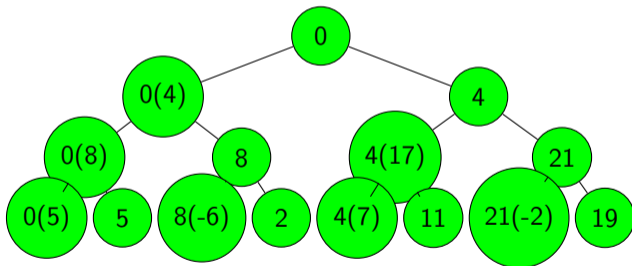
Với $d = 0$, bước nhảy của 2^{d+1} là 2

- $i = 0$, $\text{temp} = a[0 + 2^0 - 1] = a[0] = 5$
 $a[0 + 2^0 - 1] = a[0 + 2^{0+1} - 1] \rightarrow a[0] = a[1] = 0$
 $a[0 + 2^{0+1} - 1] += \text{temp} \rightarrow a[1] += \text{temp} \rightarrow a[1] = 5$
- $i = 2$, $\text{temp} = a[2 + 2^0 - 1] = a[2] = -6$
 $a[2 + 2^0 - 1] = a[2 + 2^{0+1} - 1] \rightarrow a[2] = a[3] = 8$
 $a[2 + 2^{0+1} - 1] += \text{temp} \rightarrow a[3] += \text{temp} \rightarrow a[3] = 2$
- $i = 4$, $\text{temp} = a[4 + 2^0 - 1] = a[4] = 7$
 $a[4 + 2^0 - 1] = a[4 + 2^{0+1} - 1] \rightarrow a[4] = a[5] = 4$
 $a[4 + 2^{0+1} - 1] += \text{temp} \rightarrow a[5] += \text{temp} \rightarrow a[5] = 11$
- $i = 6$, $\text{temp} = a[6 + 2^0 - 1] = a[6] = -2$
 $a[6 + 2^0 - 1] = a[6 + 2^{0+1} - 1] \rightarrow a[6] = a[7] = 21$
 $a[6 + 2^{0+1} - 1] += \text{temp} \rightarrow a[7] += \text{temp} \rightarrow a[7] = 19$

Lúc này, a trở thành $(0, 5, 8, 2, 4, 11, 21, 19)$.

Ví dụ thuật toán duyệt song song

Ở trường hợp Down sweep, sau khi đã có được một cây nhị phân hoàn hảo từ bước Up sweep, chúng ta sẽ thay thế nút gốc của cây này bằng một phần tử trung lập là 0 và các nút bên trong sẽ được tính theo thứ tự duyệt từ trên xuống dưới mà trong đó $pre[Left[u]] = pre[u]$ và $pre[Right[u]] = sum[Left[u]] + pre[u]$ với u là nút và $pre[v]$ là giá trị của nút v có được ở bước Up sweep.



Hình 6: Down sweep pass

Ví dụ thuật toán duyệt song song

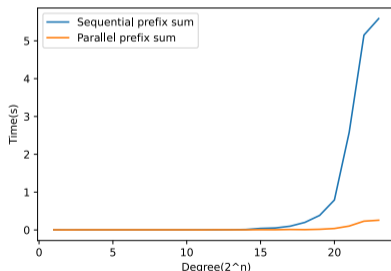
Sau khi chạy xong bước Up sweep và Down sweep, ở bước cuối cùng chúng ta sẽ tiến hành lấy danh sách a mới là $(0, 5, 8, 2, 4, 11, 21, 19)$ cộng với danh sách a ban đầu là $(5, 3, -6, 2, 7, 10, -2, 8)$ thì ta sẽ có được danh sách tổng tiền tố cho các phần tử a_k trong danh sách a ban đầu là $(5, 8, 2, 4, 11, 21, 19, 27)$.

Ở thuật toán trên, nó chỉ tính tổng tiền tố không nghịch đảo. Tuy nhiên, chúng ta có khả năng tính tổng tiền tố nghịch đảo bằng cách đảo ngược danh sách đầu vào trước khi chạy thuật toán duyệt song song và đảo ngược đầu ra sau khi đã chạy xong.

Dựa trên thuật toán này, tác giả sẽ định nghĩa ra các toán tử liên kết hai ngôi và các phần tử tương ứng đến hai thuật toán sum-product và max-product và sử dụng thuật toán duyệt song song để song song hóa quá trình tính toán nhằm giảm thời gian so với các thuật toán gốc.

So sánh thời gian thuật toán duyệt tuần tự với song song

Để so sánh sự hiệu quả của thuật toán duyệt tuần tự với song song, chúng ta sẽ tiến hành tạo ra các mảng ngẫu nhiên với kích thước là 2^n với $n = (1, \dots, 24)$. Nhìn vào Hình. 7 chúng ta có thể thấy được khi kích thước mảng dưới 2^{15} thì hầu như không có sự chênh lệch về thời gian nhưng khi kích thước mảng càng lớn thì chúng ta sẽ thấy rõ sự chênh lệch về thời gian giữa hai thuật toán này.



Hình 7: So sánh thời gian thuật toán duyệt tuần tự với song song.

Cơ sở I

Ta định nghĩa một phần tử $a_{i,k}$ một cách đệ quy như sau:

$$a_{0:1} = \psi_1(x_1) \triangleq \psi_{0,1}(x_0, x_1), \quad (13a)$$

$$a_{k-1:k} = \psi_k(x_{k-1}, x_k) \triangleq \psi_{k-1,k}(x_{k-1}, x_k), \quad (13b)$$

$$a_{T:T+1} = 1 \triangleq \psi_{T,T+1}(x_T, x_{T+1}). \quad (13c)$$

Toán tử hai ngôi kết hợp cho thuật toán thẳng-ngược (forward-backward) của một mô hình HMM với $0 \leq i < j < k$

$$a_{i:j} \otimes a_{j:k} = \sum_{x_j} \psi_{i,j}(x_i, x_j) \psi_{j,k}(x_j, x_k) \quad (14)$$

Từ biểu thức 14, một phần tử tổng quát có thể được viết như sau:

$$a_{i:k} = \psi_{i,k}(x_i, x_k) \quad (15)$$

Cơ sở II

Bổ đề 1 cho phép xây dựng thuật toán thẳng-ngược dựa trên các toán tử kết hợp

Bổ đề 1

Toán tử \otimes là kết hợp.

Định lý 1: Biểu thức tiềm năng thẳng (forward potentials)

$$a_{0:k} = \psi_{1,k}^f(x_k), k > 0. \quad (16)$$

Định lý 2: Biểu thức tiềm năng ngược (backward potentials)

$$a_{k:T+1} = \psi_{k,T}^b(x_k), k > 0. \quad (17)$$

Phân phối biên (marginal distribution) trở thành:

$$p(x_k) = \frac{1}{Z_k} a_{0:k} a_{k:T+1}. \quad (18)$$

Ý tưởng toán học

Quá trình tính toán của tiềm năng thẳng (tiềm năng ngược) tương đương với quá trình tính toán tổng tất cả tiền tố (all-prefix-sum) với toán tử hai ngôi kết hợp \otimes và phần tử $a_{i,i+1}$

$$a_{0:1} = \psi_{1,1}^f(x_1), \quad (19a)$$

$$a_{0:2} = a_{0:1} \otimes a_{1:2} = \psi_{1,2}^f(x_2), \quad (19b)$$

$$\dots \quad (19c)$$

$$a_{0:T} = a_{0,1} \otimes \dots \otimes a_{T-1:T} = \psi_{1,T}^f(x_T). \quad (19d)$$

→ Ta có thể sử dụng thuật toán duyệt song song để tính toán tiềm năng thẳng và tiềm năng ngược cùng với phân phối biên trong một quá trình song song. Với toán tử kết hợp, ta có thể thay đổi thứ tự tính toán bằng cách kết hợp các toán tử \Rightarrow chìa khóa

Thuật toán tổng-tích song song

Input : Các hàm tiềm năng $\psi_k(\cdot)$ for $k = 1 \dots T$ và toán tử \otimes .

Output: Phân phối biên $p(x_k)$ với $k = 1 \dots T$.

for $k \leftarrow 1$ **to** T **do**

▷ Chạy song song

| Thực hiện khởi tạo $a_{k-1:k}$

end

Chạy duyệt song song để tính toán $a_{0:k} = \psi_{1,k}^f(x_k)$ với $k = 1 \dots T$

for $k \leftarrow 1$ **to** T **do**

▷ Chạy song song

| Thực hiện khởi tạo $a_{k:k+1}$

end

Chạy duyệt song song nghịch đảo để tính toán $a_{k:T+1} = \psi_{k,T}^b(x_k)$ với
 $k = 1 \dots T$

for $k \leftarrow 1$ **to** T **do**

▷ Chạy song song

| $p(x_k) = \frac{1}{Z_k} a_{0:k} a_{k:T+1}$

end

Algorithm 4: Thuật toán tổng-tích song song.

Ví dụ về Thuật toán tổng-tích song song

Tình trạng tình yêu ở một thành phố được biểu diễn như một xích Markov rời rạc với tập các trạng thái $S = \{\text{'Đang yêu': } 0, \text{'Cô đơn': } 1\}$, chuỗi quan sát được $O = \{\text{'Vui vẻ': } 0, \text{'Bi quan': } 1\}$. Cho trước phân phối ban đầu $Y = \{\text{'Đang yêu': } 0.5, \text{'Cô đơn': } 0.5\}$, các ma trận chuyển trạng thái và quan sát được như sau:

| | Đang yêu | Cô đơn |
|-----------------|-----------------|---------------|
| Đang yêu | 0.65 | 0.35 |
| Cô đơn | 0.75 | 0.25 |

Bảng 8: Ma trận chuyển trạng thái.

| | Vui vẻ | Bi quan |
|-----------------|---------------|----------------|
| Đang yêu | 0.2 | 0.8 |
| Cô đơn | 0.3 | 0.7 |

Bảng 9: Ma trận quan sát được.

Sử dụng thuật toán tổng-tích song song để tính phân phối biên $p(x_k)$ và rút ra nhận xét.

Ví dụ về Thuật toán tổng-tích song song

Đặt $T = 2$

Pha khởi tạo: thực hiện khởi tạo $a_{k-1:k}$ với $k = 1 \dots T$ một cách đệ quy như sau:

$$a_{0:1} = \psi_1(x_1) = p(y_1|x_1)p(x_1) = \begin{bmatrix} 0.2 & 0.3 \end{bmatrix} \times \begin{bmatrix} 0.5 & 0.5 \end{bmatrix} = \begin{bmatrix} 0.1 & 0.15 \end{bmatrix}$$

$$a_{1:2} = \psi_2(x_1, x_2) = \begin{bmatrix} 0.8 & 0.7 \end{bmatrix} \cdot \begin{bmatrix} 0.65 & 0.35 \\ 0.75 & 0.25 \end{bmatrix} = \begin{bmatrix} 0.52 & 0.245 \\ 0.6 & 0.175 \end{bmatrix}$$

Sử dụng duyệt song song, tính được:

$$a_{0:k} = \psi_{1,k}^f(x_k) = a_{0:1} \cdot a_{1:2} = \begin{bmatrix} 0.1 & 0.15 \\ 0.142 & 0.05075 \end{bmatrix}$$

Pha khởi tạo: thực hiện khởi tạo $a_{k:k+1}$ với $k = 1 \dots T$ một cách đệ quy như sau:

$$a_{1:2} = \psi_2(x_1, x_2) = \begin{bmatrix} 0.8 & 0.7 \end{bmatrix} \cdot \begin{bmatrix} 0.65 & 0.35 \\ 0.75 & 0.25 \end{bmatrix} = \begin{bmatrix} 0.52 & 0.245 \\ 0.6 & 0.175 \end{bmatrix}$$

$$a_{2:3} = a_{2:2} = \begin{bmatrix} 1.0 & 1.0 \\ 1.0 & 1.0 \end{bmatrix}$$

Ví dụ về Thuật toán tổng-tích song song (Tiếp)

Sử dụng duyệt song song nghịch đảo, tính được:

$$a_{k:T+1} = \psi_{k,T}^b(x_k) = \begin{bmatrix} 0.765 & 0.775 \\ 1.0 & 1.0 \end{bmatrix}$$

Nhờ việc tính toán song song, ta đã tính toán được hàm tiềm năng thẳng và tiềm năng ngược một cách nhanh chóng. Bước cuối cùng là tính toán phân phối biên, kết quả thu được như sau:

$$p(x_k) = \frac{1}{Z_k} a_{0:k} a_{k:T+1} = \frac{\begin{bmatrix} 0.1 & 0.15 \end{bmatrix} \times \begin{bmatrix} 0.765 & 0.775 \end{bmatrix}}{\sum \begin{bmatrix} 0.1 & 0.15 \end{bmatrix} \times \begin{bmatrix} 0.765 & 0.775 \end{bmatrix}} +$$

$$\frac{\begin{bmatrix} 0.142 & 0.05075 \end{bmatrix} \times \begin{bmatrix} 1.0 & 1.0 \end{bmatrix}}{\sum \begin{bmatrix} 0.142 & 0.05075 \end{bmatrix} \times \begin{bmatrix} 1.0 & 1.0 \end{bmatrix}} = \begin{bmatrix} 0.39688716 & 0.60311284 \\ 0.73670558 & 0.26329442 \end{bmatrix}$$

Vậy với chuỗi các quan sát $O = (\text{'Vui vẻ': } 0, \text{'Bi quan': } 1)$ thì kết quả dự đoán là $\{\text{'Cô đơn', 'Đang yêu'}\}$

Phân tích độ phức tạp thuật toán

Sử dụng hai khái niệm work và span để phân tích độ phức tạp của một thuật toán song song.

Mệnh đề 1.a) Thuật toán tổng-tích song song có span complexity là $O(\log T)$

- Giai đoạn khởi tạo: $O(1)$.
- Giai đoạn chạy duyệt song song: $O(\log T)$.

Mệnh đề 1.b) Thuật toán tổng tích song song có work complexity là $O(T)$

- Giai đoạn khởi tạo: $O(T)$.
- Giai đoạn chạy duyệt song song: $O(T)$.

Ý tưởng của thuật toán Viterbi cổ điển

Thuật toán viterbi dựa trên lập trình động, mục đích tính giá trị ước lượng $x_{1:T}^*$ bằng cách cực đại phân phối hậu nghiệm (posterior distribution).

$$p(x_{1:T}|y_{1:T}) = \frac{p(y_{1:T}, x_{1:T})}{p(y_{1:T})} \propto p(y_{1:T}, x_{1:T})$$

Tương đương việc cực đại phân phối của xác suất hậu nghiệm:

$$p(y_{1:T}, x_{1:T}) = p(x_1)p(y_1|x_1) \prod_{t=2}^T p(y_t|x_t)p(x_t|x_{t-1})$$

Tiếp theo, ta thực hiện lấy vị trí cực đại của phân phối xác suất hậu nghiệm này như sau:

$$x_{1:T}^* = \arg \max_{x_{1:T}} p(y_{1:T}, x_{1:T})$$

Thuật toán Viterbi cổ điển

Giả sử $V_{k-1}(x_{k-1})$ là xác suất cực đại của $x_{1:k-1}^*$ kết thúc tại trạng thái x_{k-1} và trạng thái tối ưu x_{k-1}^* là một hàm của x_k được kí hiệu dưới dạng $u_{k-1}(x_k)$.

$$V_k(x_k) = \max_{x_{k-1}} [p(y_k|x_k)p(x_k|x_{k-1})V_{k-1}(x_{k-1})],$$

$$u_{k-1}(x_k) = \arg \max_{x_{k-1}} [p(y_k|x_k)p(x_k|x_{k-1})V_{k-1}(x_{k-1})].$$

Bằng cách cực đại phân phối xác suất hậu nghiệm để ước lượng $x_{1:T}^*$ ta được:

$$x_T^* = \arg \max_{x_T} (V_T(x_T)).$$

Với điều kiện khởi tạo và đường dẫn Viterbi $x_{1:T}^*$ tính từ forward và backward:

$$V_1(x_1) = p(x_1)p(y_1|x_1),$$

$$x_{k-1}^* = u_{k-1}(x_k^*).$$

Mã giả của thuật toán Viterbi cổ điển

Input : Các hàm tiềm năng $\psi_k(\cdot)$ với $k = 1, \dots, T$.

Output: Đường dẫn Viterbi $x_{1:T}^*$.

//Forward pass:

$$V_1(x_1) = \psi_1(x_1)$$

for $k \leftarrow 2$ **to** T **do**

▷ Chạy tuần tự

$$\left| \begin{array}{l} \overline{V}_k(x_k) = \max_{x_{k-1}} [p(y_k|x_k)p(x_k|x_{k-1})V_{k-1}(x_{k-1})] \\ u_{k-1}(x_k) = \arg \max_{x_{k-1}} [p(y_k|x_k)p(x_k|x_{k-1})V_{k-1}(x_{k-1})] \end{array} \right.$$

end

//Backward pass:

$$x_T^* = \arg \max_{x_T} V_T(x_T)$$

for $k \leftarrow T$ **to** 2 **do**

$$\left| x_{k-1}^* = u_{k-1}(x_k^*) \right.$$

end

Algorithm 5: Thuật toán Viterbi cổ điển.

Phân tích độ phức tạp

Độ phức tạp tính toán ở giai đoạn forward là: $O(D^2T)$.

Độ phức tạp tính toán ở giai đoạn backward là: $O(T)$.

Độ phức tạp của thuật toán Viterbi là: $O(D^2T)$.

Ví dụ cho thuật toán viterbi cổ điển

Vấn đề của khá [4], khá phân tích các cảm xúc của bạn nữ trước khi nhận được thư của khá bao gồm $S = \{\text{'Không vui'} : 0, \text{'Vui'} : 1\}$ và khá coi quan sát các thư hồi âm bao gồm $O = \{\text{'Không trả lời'}, \text{'Trả lời'}\}$. Khá đặt xác suất cảm xúc của cô gái này là $I = \{\text{'Không vui'} : 0.5, \text{'Vui'} : 0.5\}$. Dựa trên các quan sát của khá, khá lập được 2 bảng sau:

| | Không vui | Vui |
|-----------|-----------|-----|
| Không vui | 0.6 | 0.4 |
| Vui | 0.8 | 0.2 |

Bảng 10: Ma trận chuyển trạng thái.

| | Không trả lời | Trả lời |
|-----------|---------------|---------|
| Không vui | 0.9 | 0.1 |
| Vui | 0.3 | 0.7 |

Bảng 11: Ma trận quan sát được.

Chúng ta sẽ giúp khá phân tích quy luật thay đổi các trạng thái cảm xúc của bạn nữ bằng thuật toán viterbi cổ điển để khá nhận được hồi âm sau khi gửi thư.

Ví dụ về thuật toán viterbi cổ điển (tiếp)

Pha khởi tạo (forward): với $k = 1$

$$V_1(x_1) = \psi_1(x_1) = p(x_1)p(y_1|x_1) = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix} \times \begin{bmatrix} 0.9 & 0.3 \end{bmatrix} = \begin{bmatrix} 0.45 & 0.15 \end{bmatrix}$$

Thực hiện vòng lặp tính cho $k = 2$ (forward):

Bước 1: tính giá trị $\psi_k(x_{k-1}, x_k)V_{k-1}(x_{k-1})$

$$\Leftrightarrow p(y_k|x_k)p(x_k|x_{k-1})V_{k-1}(x_{k-1}) \Leftrightarrow p(y_2|x_2)p(x_2|x_1)V_1(x_1)(*):$$

$$\psi_2(x_1, x_2) = p(y_2|x_2)p(x_2|x_1) = \begin{bmatrix} [0.1] & [0.7] \end{bmatrix} \times \begin{bmatrix} [0.6 & 0.8] \\ [0.4 & 0.2] \end{bmatrix} = \begin{bmatrix} [0.06 & 0.08] \\ [0.28 & 0.14] \end{bmatrix}$$

sau đó tính giá trị của biểu thức (*):

$$\psi_2(x_1, x_2)V_1(x_1) = \begin{bmatrix} [0.06 & 0.08] \\ [0.28 & 0.14] \end{bmatrix} \times \begin{bmatrix} 0.45 & 0.15 \end{bmatrix} = \begin{bmatrix} [0.027 & 0.012] \\ [0.126 & 0.021] \end{bmatrix}$$

Ví dụ thuật toán viterbi cổ điển (tiếp)

Bước 2: tính giá trị $V_k(x_k)$ tương đương max của biểu thức (*):

$$V_2(x_2) = [0.027 \quad 0.126]$$

Bước 3: tính $u_{k-1}(x_k)$ tương đương vị trí cực đại của biểu thức (*):

$$u_1(x_2) = [0 \quad 0]$$

Đặt kết quả trạng thái cuối là vị trí cực đại của $V_T(x_T)$ (backward):

$$x_2^* = \arg \max_{x_2} V_T(x_T) = \arg \max ([0.027 \quad 0.126]) = 1$$

Tương tự cho trạng thái đầu là:

$$x_1^* = \arg \max_{x_1} ([0.45 \quad 0.15]) = 0$$

Vậy với chuỗi quan sát là $O = \{\text{Không trả lời} : 0, \text{Trả lời} : 1\}$ thì kết quả dự đoán là: $\{\text{'Không vui'}, \text{'Vui'}\}$. Kết quả này giống với mong đợi của khách.

Song song hoá dựa trên đường dẫn

Đặt $\tilde{a}_{i:j}$ bao gồm một cặp đường dẫn tối ưu từ trạng thái x_i đến x_j , và xác suất tối đại của đường dẫn đó. Ta được:

$$\tilde{a}_{i:j} = \begin{pmatrix} A_{i:j}(x_i, x_j) \\ \hat{X}_{i:j}(x_i, x_j) \end{pmatrix}, \tilde{a}_{j:k} = \begin{pmatrix} A_{j:k}(x_j, x_k) \\ \hat{X}_{j:k}(x_j, x_k) \end{pmatrix} \quad (20)$$

Khi đó toán tử hai ngôi kết hợp \vee cho ước lượng xác suất hậu nghiệm cực đại là:

$$\tilde{a}_{i:k} = \tilde{a}_{i:j} \vee \tilde{a}_{j:k}.$$

Cơ sở III

Bổ đề 2

Toán tử \vee là kết hợp.

Định lý 3:

$$A_{i:j}(x_i, x_j) = \max_{x_{i+1:j-1}} \prod_{k=i+1}^j \psi_{k-1,k}(x_{k-1}, x_k),$$

$$X_{i:j}(x_i, x_j) = \arg \max_{x_{i+1:j-1}} \prod_{k=i+1}^j \psi_{k-1,k}(x_{k-1}, x_k).$$

Cơ sở IV

Hệ quả

$$\tilde{a}_{0:T+1} = \left(\psi_1(x_1^*) \prod_{t=2}^T \psi_{t-1,t}(x_{t-1}^*, x_t^*) \right),$$

Trong đó, $x_{1:T}^*$ là xác suất tối đại hậu nghiệm.

Sau khi đã có được các thành phần cần thiết cho quá trình song song nhưng việc lưu trữ các $\tilde{a}_{i:j}$ tốn chi phí bộ nhớ cao, từ đó tác giả đề xuất sử dụng max-product để giảm chi phí lưu trữ.

Công thức max-product

Đặt $\tilde{\psi}_k^b(x_k)$, $\tilde{\psi}_k^f(x_k)$ lần lượt là xác suất tối đại của đường dẫn bắt đầu và kết thúc tại x_k . Ta có:

$$\begin{aligned}\tilde{\psi}_k^f(x_k) &= A_{0:k}(x_0, x_k), \\ \tilde{\psi}_k^b(x_k) &= A_{k:T+1}(x_k, x_{T+1}).\end{aligned}$$

Từ định nghĩa (20) ta suy ra được bổ đề 3 và sử dụng đệ qui để tính các giá trị đó.

Cơ sở V

Bổ đề 3: Xác suất cực đại của forward và backward

$$\tilde{\psi}_k^f(x_k) = \max_{x_{k-1}} \psi_{k-1:k}(x_{k-1}, x_k) \tilde{\psi}_{k-1}^f(x_{k-1}),$$

$$\tilde{\psi}_k^b(x_k) = \max_{x_{k+1}} \psi_{k:k+1}(x_k, x_{k+1}) \tilde{\psi}_{k+1}^b(x_{k+1}).$$

Điều kiện khởi tạo:

$$\tilde{\psi}_1^f(x_1) = \psi_1(x_1),$$

$$\tilde{\psi}_T^b(x_T) = 1.$$

Bằng cách cực đại $\tilde{\psi}_k^f(x_k) \tilde{\psi}_k^b(x_k)$ ta ước lượng được giá trị phân phối tối đại hậu nghiệm tại bước thứ k như sau:

$$x_k^* = \arg \max_{x_k} \tilde{\psi}_k^f(x_k) \tilde{\psi}_k^b(x_k), k = 1, \dots, T.$$

Cơ sở VI

Đặt $\bar{a}_{i:j}$ là xác suất tối đại của nằm trong định nghĩa $\tilde{a}_{i:j}$ nhằm mục đích tránh việc lưu trữ đường dẫn, giảm chi phí lưu trữ.

Từ định nghĩa trên kết hợp với toán tử \vee ta được:

$$\bar{a}_{i:k} = \bar{a}_{i:j} \vee \bar{a}_{j:k},$$

với

$$\bar{a}_{i:k} = \max_{x_j} A_{i:j}(x_i, x_j) A_{j:k}(x_j, x_k).$$

Hàm tiềm năng thẳng tối đại được tính như sau:

$$\bar{a}_{0:k} = \tilde{\psi}_k^f(x_k), k > 0.$$

Hàm tiềm năng ngược tối đại được tính như sau:

$$\bar{a}_{k:T+1} = \tilde{\psi}_k^b(x_k), k > 0.$$

Mã giả của thuật toán song song max-product

Input : Các hàm tiềm năng $\psi_k(\cdot)$ for $k = 1 \dots T$ và toán tử \vee .

Output: Đường dẫn Viterbi $x_{1:T}^*$.

for $k \rightarrow 1$ **to** T **do**

▷ Chạy song song

| Thực hiện khởi tạo $\bar{a}_{k-1:k}$

end

Chạy duyệt song song để tính toán $\bar{a}_{0:k} = \tilde{\psi}_{1,k}^f(x_k)$ với $k = 1 \dots T$

for $k \rightarrow 1$ **to** T **do**

▷ Chạy song song

| Thực hiện khởi tạo $\bar{a}_{k:k+1}$

end

Chạy duyệt song song nghịch đảo để tính toán $\bar{a}_{k:T+1} = \tilde{\psi}_{k,T}^b(x_k)$ với $k = 1 \dots T$

for $k \rightarrow 1$ **to** T **do**

▷ Chạy song song

| Tính x_k^* sử dụng công thức ở Cơ sở V

end

Algorithm 6: Thuật toán song song max-product.

Phân tích thời gian chạy thuật toán song song max-product

Độ phức tạp **span** là $O(\log T)$:

- Ở giai đoạn khởi tạo và kết hợp ở vòng lặp cuối là: $O(1)$.
- Thuật toán duyệt song song: $O(\log T)$.

Độ phức tạp **work** là $O(T)$:

- Ở giai đoạn khởi tạo và kết hợp ở vòng lặp cuối là: $O(T)$.
- Thuật toán duyệt song song: $O(T)$.

Ví dụ cho thuật toán song song max-product

Ta cũng lấy ví dụ của khá và người yêu để làm ví dụ ở đây, nhưng ta sử dụng thuật toán song song max-product để cải thiện tốc độ tính toán mục đích cho khá biết được kết quả nhanh nhất.

Pha khởi tạo: thực hiện khởi tạo $\bar{a}_{k-1:k}$ với $k = 1 \dots T$ (forward):

$$\bar{a}_{0:1} = \psi_1(x_1) = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix} \times \begin{bmatrix} 0.9 & 0.3 \end{bmatrix} = \begin{bmatrix} 0.45 & 0.15 \end{bmatrix}$$

$$\bar{a}_{1:2} = A_{1:2}(x_1, x_2) = \psi_2(x_1, x_2) = \begin{bmatrix} 0.1 & 0.7 \end{bmatrix} \times \begin{bmatrix} 0.6 & 0.4 \\ 0.8 & 0.2 \end{bmatrix} = \begin{bmatrix} 0.06 & 0.28 \\ 0.08 & 0.14 \end{bmatrix}$$

$$\bar{a}_{0:2} = \bar{a}_{0:1} \vee \bar{a}_{1:2} = \max_{x_1} \left(\psi_{1,2}(x_1, x_2) \tilde{\psi}_1^f(x_1) \right) = \max_{x_1} \left(\psi_2(x_1, x_2) \psi_1(x_1) \right)$$

$$= \max_{x_1} \left(\begin{bmatrix} 0.027 & 0.012 \\ 0.126 & 0.021 \end{bmatrix} \right) = \begin{bmatrix} 0.027 & 0.126 \end{bmatrix}$$

Ví dụ cho thuật toán song song max-product (tiếp)

Chạy duyệt song song đảo để tính giá trị $\bar{a}_{k:T+1}$ với $k = 1, \dots, T$:

- $k = T = 2$

$$\bar{a}_{T:T+1} = \tilde{\psi}_T^b(x_T) = \tilde{\psi}_2^b(x_2) = 1$$

- $k = 1$

$$\bar{a}_{k:T+1} = \tilde{\psi}_k^b(x_k) = \max_{x_{k+1}} \left(\psi_{k,k+1}(x_k, x_{k+1}) \tilde{\psi}_{k+1}^b(x_{k+1}) \right)$$

$$\Leftrightarrow \bar{a}_{1:T+1} = \tilde{\psi}_1^b(x_1) = \max_{x_2} \left(\psi_{1,2}(x_1, x_2) \tilde{\psi}_2^b(x_2) \right) = \max_{x_2} (\psi_2(x_1, x_2))$$

$$= \max_{x_2} \left(\begin{bmatrix} [0.06 & 0.28] \\ [0.08 & 0.14] \end{bmatrix} \right) = [0.28 \quad 0.14]$$

Ví dụ cho thuật toán song song max-product (tiếp)

Tính x_k^* bằng cách lấy vị trí cực đại của biểu thức $\tilde{\psi}_k^f(x_k)\tilde{\psi}_k^b(x_k)$ với $k = 1, \dots, T$.

$$x_1^* = \arg \max_{x_1} \left(\tilde{\psi}_1^f(x_1)\tilde{\psi}_1^b(x_1) \right) = \arg \max_{x_1} ([0.45 \quad 0.15] \times [0.28 \quad 0.14])$$

$$= \arg \max_{x_2} ([0.126 \quad 0.021]) = 0$$

$$x_2^* = \arg \max_{x_2} \left(\tilde{\psi}_2^f(x_2)\tilde{\psi}_2^b(x_2) \right) = \arg \max_{x_2} ([0.027 \quad 0.126] \times 1) = 1$$

Vậy với chuỗi quan sát là $O = \{\text{Không trả lời} : 0, \text{Trả lời} : 1\}$ thì kết quả dự đoán là: $\{\text{'Không vui'}, \text{'Vui'}\}$. Kết quả cho phép khá tính nhanh hơn phương pháp trước.

Phương pháp thực nghiệm I

Sử dụng mô hình kênh Gilbert-Elliot với hai đầu vào: tín hiệu đầu vào b_k và tín hiệu nhiễu s_k , cả hai đều là tín hiệu nhị phân. Tín hiệu đầu vào b_k bị lật bằng toán tử xor bởi một lỗi phụ thuộc v_k .

$$y_k = b_k \oplus v_k. \quad (21)$$

Tín hiệu nhiễu s_k được đại diện bởi mô hình Markov ẩn hai trạng thái, "good" hoặc "bad". Trạng thái "good" có xác suất lỗi thấp, còn trạng thái "bad" có xác suất lỗi cao. Ta gọi:

- q_0 : đại diện cho xác suất của lỗi trong trạng thái "good".
- q_1 : đại diện cho xác suất của lỗi trong trạng thái "bad".

Ma trận chuyển xác suất được đại diện với:

- p_0 là xác suất chuyển từ trạng thái độ lỗi cao (bad) đến trạng thái độ lỗi thấp (good).

Phương pháp thực nghiệm II

- p_1 là xác suất chuyển từ trạng thái độ lỗi thấp (good) đến trạng thái độ lỗi cao (bad).
- p_2 là xác suất chuyển trạng thái của b_k .

Sử dụng một mô hình kết $x_k = (s_k, b_k)$ - xích Markov 4 trạng thái để phục hồi các trạng thái ẩn:

$$x_k = \{(0, 0), (0, 1), (1, 0), (1, 1)\} \longrightarrow \text{encode: } x_k = \{0, 1, 2, 3\}. \quad (22)$$

Ma trận chuyển trạng thái (transition matrix) mã hóa thông tin trong $p(x_k | x_{k-1})$.

$$\Pi = \begin{pmatrix} (1-p_0)(1-p_2) & p_0(1-p_2) & (1-p_0)p_2 & p_0p_2 \\ p_1(1-p_2) & (1-p_1)(1-p_2) & p_1p_2 & (1-p_1)p_2 \\ (1-p_0) & p_0p_2 & (1-p_0)(1-p_2) & p_0(1-p_2) \\ p_1p_2 & (1-p_1)p_2 & p_1(1-p_2) & (1-p_1)(1-p_2) \end{pmatrix} \quad (23)$$

Phương pháp thực nghiệm III

Ma trận quan sát (observation matrix) mã hóa thông tin trong $p(y_k | x_k)$.

$$O = \begin{pmatrix} (1 - q_0) & q_0 \\ (1 - q_1) & q_1 \\ q_0 & (1 - q_0) \\ q_1 & (1 - q_1) \end{pmatrix} \quad (24)$$

Phương pháp đánh giá và cài đặt siêu tham số

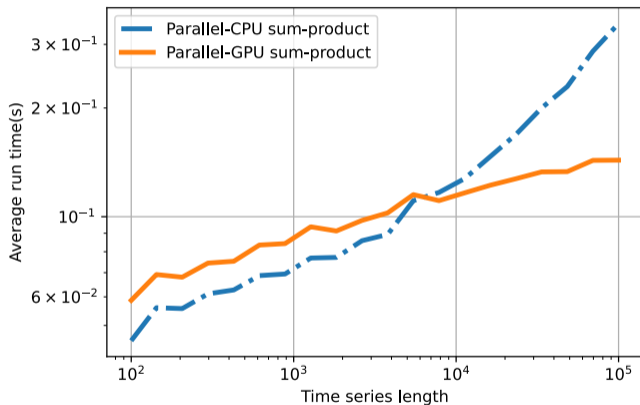
Phương pháp đánh giá cho mô hình đề xuất:

- Dữ liệu đánh giá: Các trạng thái và các phép đo được mô phỏng với các chiều dài chuỗi thời gian khác nhau từ $T = 10^2$ đến $T = 10^5$.
- Tính toán kết quả: Lấy trung bình số lần chạy. Đối với các phương pháp tuần tự, sử dụng 10 lần chạy. Với các phương pháp song song, sử dụng 100 lần chạy.
- So sánh kết quả: Thực hiện đánh giá về mặt thời gian thực thi thuật toán, không đánh giá hiệu suất lỗi bởi vì các phương pháp song song và tuần tự là tương đương về mặt đại số và do đó, không có sự khác biệt về hiệu suất lỗi của chúng.

Các siêu tham số của mô hình Gilbert–Elliott trong biểu thức 23 như sau:

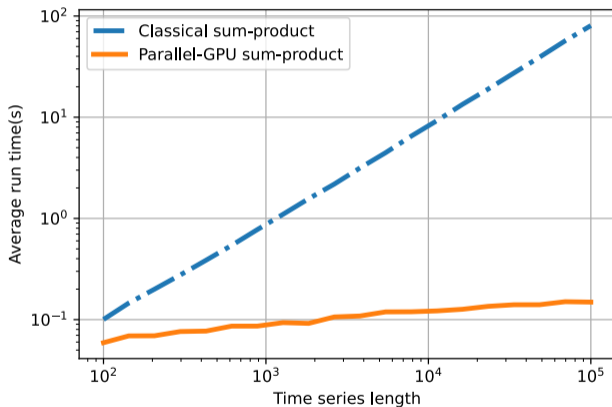
$p_0 = 0.03, p_1 = 0.1, p_2 = 0.05, q_0 = 0.01, q_1 = 0.1$. Xác suất khởi tạo cho bốn trạng thái: $p(x_1) = 0.25$ với $x_1 \in \{0, 1, 2, 3\}$

So sánh thời gian parallel sum-product trên CPU và GPU



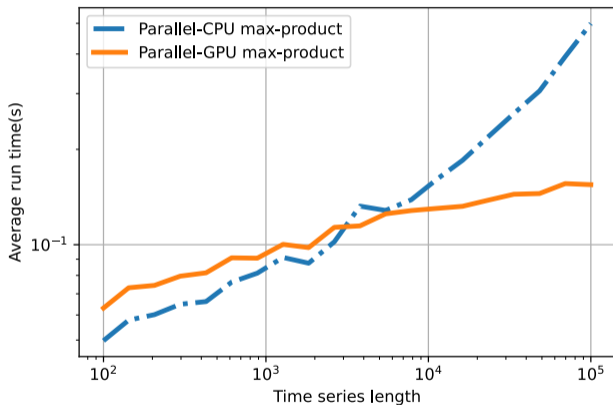
Hình 8: So sánh thời gian parallel sum-product trên CPU và GPU.

So sánh thời gian classical và parallel sum-product



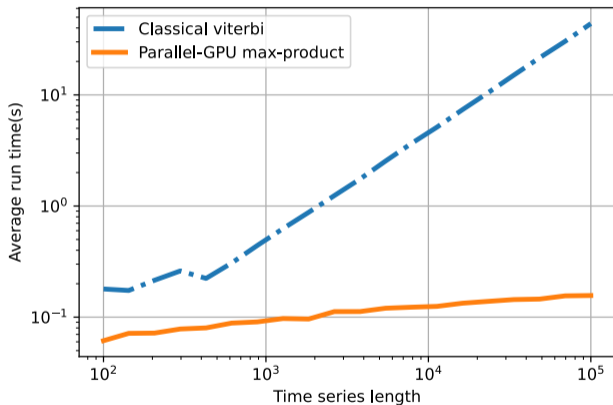
Hình 9: So sánh thời gian classical và parallel sum-product.

So sánh thời gian parallel max-product trên CPU và GPU



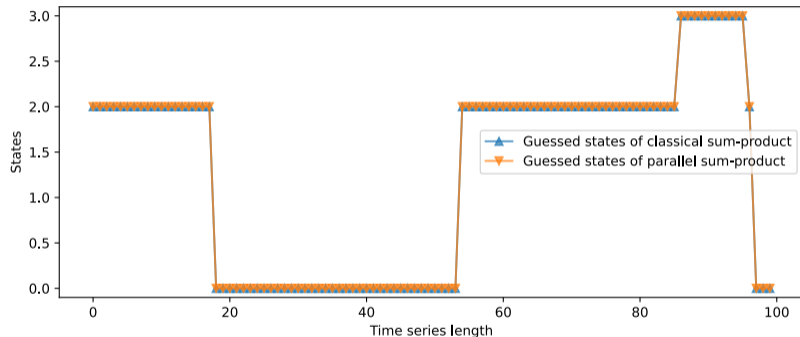
Hình 10: So sánh thời gian parallel max-product trên CPU và GPU.

So sánh thời gian classical viterbi và parallel max-product



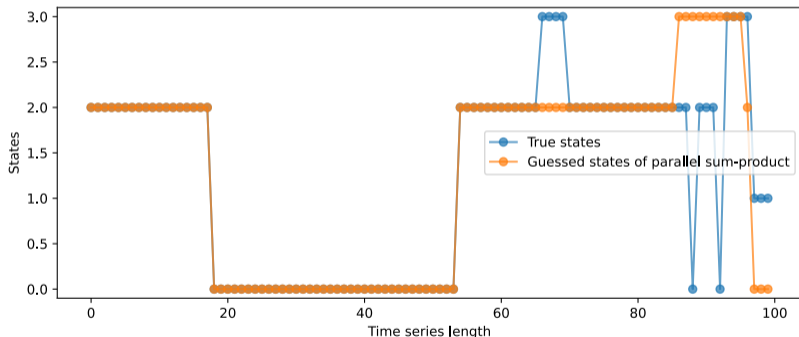
Hình 11: So sánh thời gian classical viterbi và parallel max-product.

So sánh kết quả suy diễn classical và parallel sum-product



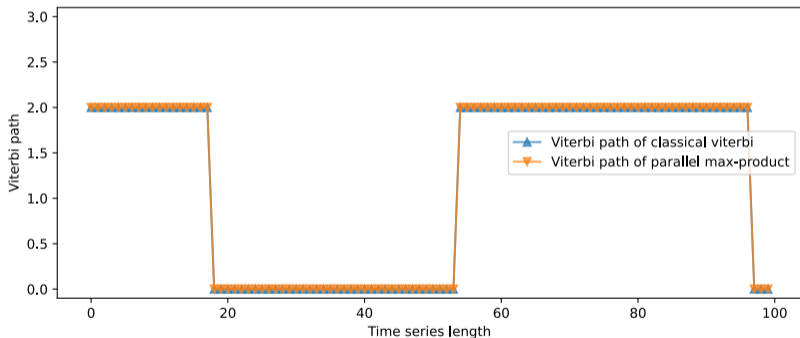
Hình 12: So sánh kết quả suy diễn classical và parallel sum-product.

So sánh kết quả suy diễn parallel sum-product với trạng thái đúng



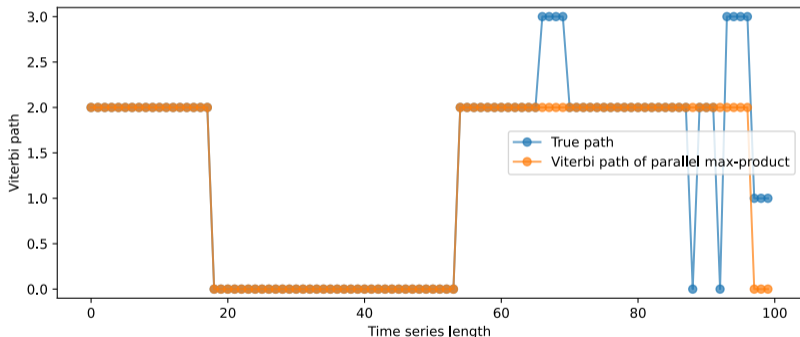
Hình 13: So sánh kết quả suy diễn parallel sum-product với trạng thái đúng.

So sánh đường dẫn viterbi giữa classical viterbi và parallel max-product



Hình 14: So sánh đường dẫn viterbi classical viterbi và parallel max-product.

So sánh đường dẫn viterbi của parallel max-product với đường dẫn đúng






Hình 15: So sánh đường dẫn viterbi của parallel max-product với đường dẫn đúng.

Kết luận

- Kết hợp phân phối biên trơn (marginal smoothing distribution) và ước lượng cực đại hậu nghiệm để đưa ra công thức tính song song cho bài toán suy diễn HMM.
- Quá trình tính toán dựa trên sự kết hợp giữa các toán tử kết hợp hai ngôi và thuật toán duyệt song song như đó đã giảm thời gian tính toán từ tuyến tính xuống còn logarit.
- Chứng minh được tính hiệu quả của phương pháp đề xuất bằng các thực nghiệm trên CPU và GPU.

Tài liệu tham khảo I

-  Tanmay Binaykiya.
Hidden markov models.
<https://tanmaybinaykiya.github.io/hmm-applications>, 2018.
-  Shaunak Chatterjee and Stuart J. Russell.
A temporally abstracted viterbi algorithm.
In [UAI](#), 2011.
-  Zihui Du, Zhaoming Yin, and David A. Bader.
A tile-based parallel viterbi algorithm for biological sequence alignment on gpu with cuda.
[2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum \(IPDPSW\)](#), pages 1–8, 2010.

Tài liệu tham khảo II



Nguyễn Tiên Dũng.

Mô hình markov ẩn.

<https://jurgendn.github.io/mathematics/2020/06/16/Hidden-Markov-Model/>, 2020.



Sean R Eddy.

What is a hidden Markov model?

[Nature Biotechnology](#), 22(10):1315–1316, October 2004.



Shawn R. Hymel.

Massively parallel hidden markov models for wireless applications.
2011.

Tài liệu tham khảo III

 Jun Li, Shuangping Chen, and Yanhui Li.

The fast evaluation of hidden markov models on gpu.

[2009 IEEE International Conference on Intelligent Computing and Intelligent Systems](#), 4:426–430, 2009.

 Yury Lifshits, Shay Mozes, Oren Weimann, and Michal Ziv-Ukelson.

Speeding up hmm decoding and training by exploiting sequence repetitions.

[Algorithmica](#), 54:379–399, 2007.

 Chuang Liu.

cuhmm : a cuda implementation of hidden markov model training and classification.

2009.

Tài liệu tham khảo IV

-  Saeed Maleki, Madan Musuvathi, and Todd Mytkowicz.
Parallelizing dynamic programming through rank convergence.
In [PPoPP '14](#), 2014.
-  Jesper Sindahl Nielsen and Andreas Sand.
Algorithms for a parallel implementation of hidden markov models with a small state space.
[2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum](#), pages 452–459, 2011.
-  Mirco Ravanelli.
Deep learning for distant speech recognition.
[arXiv preprint arXiv:1712.06086](#), 2017.

Tài liệu tham khảo V



Andreas Sand, Christian N. S. Pedersen, Thomas Mailund, and Asbjorn Tolbol Brask.

Hmmlib: A c++ library for general hidden markov models exploiting modern cpus.

[2010 Ninth International Workshop on Parallel and Distributed Methods in Verification, and Second International Workshop on High Performance Computational Systems Biology, pages 126–134, 2010.](#)



Simo Särkkä and Ángel F. García-Fernández.

Temporal parallelization of bayesian smoothers.

[IEEE Transactions on Automatic Control, 66\(1\):299–306, 2021.](#)

Tài liệu tham khảo VI

-  Fatemeh Yaghoobi, Adrien Corenflos, Sakira Hassan, and Simo Särkkä. Parallel iterated extended and sigma-point kalman smoothers. In ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 5350–5354, 2021.
-  Dan Zhang, Rong cai Zhao, Lin Han, Tao Wang, and Jin Qu. An implementation of viterbi algorithm on gpu. 2009 First International Conference on Information Science and Engineering, pages 121–124, 2009.