ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Báo cáo đồ án 02

Toán ứng dụng và thống kê

Nhóm sinh viên thực hiện

Nguyễn Văn Hưng - 20120009
Trần Ngọc Đô - 20120057
Huỳnh Minh Tuấn - 20120024

Giảng viên hướng dẫn

Nguyễn Đình Thúc
Nguyễn Văn Quang Huy
Võ Nam Thục Đoan

Tháng 5 năm 2022

1 Thành phần của một mô hình Markov ẩn là gì? Chúng khác gì so với mô hình Markov

Các thành phần của mô hình Markov ẩn bao gồm:

- Tập các quan sát $O = o_1, o_2, ...o_T$. Với mỗi $o_i \in V = \{v_1, v_2, ...v_K\}$
- Tập các trạng thái $Q = q_1, q_2, ...q_N$.
- Ma trận chuyển trạng thái của các trạng thái thuộc S là $A = (a_{i,j})$.
- Ma trận xác suất phụ thuộc trạng thái (emission) $B = (b_i(o_t))$.
- Phân phối xác suất ban đầu của các trạng thái π.

Trong khi đó với mô hình Markov chỉ cần 3 thành phần:

- Tập các trạng thái $Q = q_1, q_2, ...q_N$.
- Ma trận chuyển trạng thái của các trạng thái thuộc S là $A = (a_{i,j})$.
- Phân phối xác suất ban đầu của các trạng thái π.

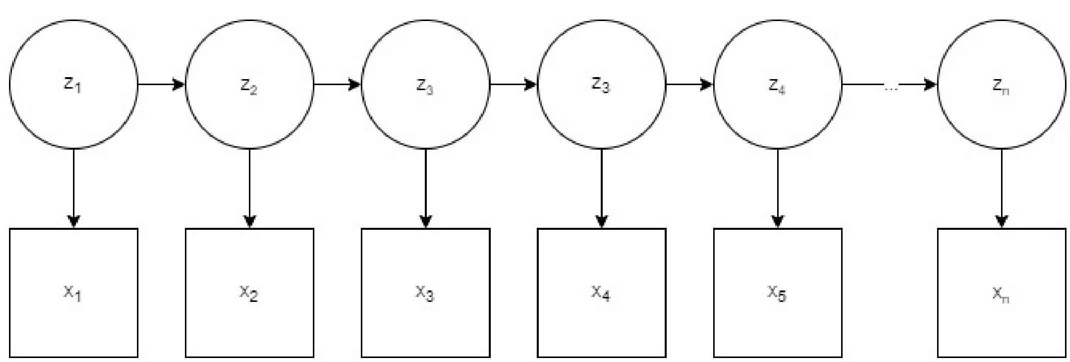
2 Các giả thiết (assumpsion) đặt ra cho mô hình Markov ẩn là gì? Tìm ví dụ các bài toán mà các giả thiết này hợp lý và bất hợp lý

Giả thiết cần cho mô hình Markov ẩn là:

- Xác suất của mỗi trạng thái hiện tại chỉ phụ thuộc vào trạng thái trạng trước đó
- Xác suất để có được quan sát o_t nào chỉ phụ thuộc vào trạng thái q_i chứ không phụ thuộc nhiều trạng thái q hay nhiều quan sát o khác.

3 Cho một mô hình Markov ẩn với các tham số đã biết, thuật toán tiến trước (forward algorithm) được dùng để xác định độ hợp lý (likelihood) của một chuỗi quan sát (observation). Mô tả và đánh giá độ phức tạo của thuật toán tiến trước

Forward algorithm là một thuật toán tối ưu cho mô hình Hidden Markov Model sử dụng kĩ thuật Dynamic Programming.



Giả sử có một mô hình Markov ẩn với M trạng thái ẩn và N trạng thái có thể quan sát. Forward algorithm được dùng để giải quyết bài toán dạng cho biết mức độ hợp l của một chuỗi Markov ẩn có độ dài T. Forward sử dụng cấu trúc dữ liệu dạng bảng để lưu lại độ hợp lí nếu trạng thái ẩn thứ j $(0 \leq j \leq M)$ xuất hiện tại thời điểm t $(1 \leq t \leq T)$ và sử dụng kĩ thuật Bottom Up để làm đầy bảng.

Với ý tưởng trên, đặt mảng $\alpha[j,t]$ là độ hợp lí nếu trạng thái ẩn thứ j xảy ra tại thời điểm t ta có mã giả của foward_algorithm:

procedure FORWARD_ALGORITHM$(a[1..N,1..N], b[1..N,1..K], \pi[1..N], o[1..T])$
    $\alpha[1..N,1..T] = \{0\}$
    $\alpha[,1] = \pi * b[,o[1]]$
    for each $t \in (2,T)$ **do**
        for each $j \in (1,N)$ **do**
            $\alpha[j,t] = \sum_{i=1}^{N} \alpha[i,t-1] * a[i,j] * b[j,o[t]]$
        end for
    end for
    $likelihood = \sum_{i=1}^{N} \alpha[i,T]$
end procedure

Chọn thao tác cơ sở là $\alpha[j,t] = \sum_{i=1}^{N} \alpha[i,t-1] * a[i,j] * b[j,o[t]]$ ta có biểu thức biểu diễn độ phức tạp thuật toán như sau:

$$Complexity = O(\sum_{t=2}^{T}\sum_{j=1}^{N} N) \approx O(N^2.T)$$

4 Cho một mô hình Markov ẩn với các tham số đã biết, thuật toán Viterbi được dùng để xác định chuỗi trạng thái (state) khả dĩ nhất. Mô tả và đánh giá độ phức tạp của thuật toán Viterbi

Viterbi algorithm tương tự như forward algorithm là một thuật toán tối ưu sử dụng kĩ thuật Dynamic Programming để giảm độ phức tạp.

Cho một mô hình Markov ẩn với M trạng thái ẩn, N trạng thái quan sát cùng các tham số liên quan cùng một chuỗi quan sát (*observations*) có độ dài T. Sử dụng Viterbi ta có thể tìm được chuỗi bị ẩn phù hợp nhất với chuỗi quan sát được cung cấp. Xét tại thời điểm t bất kì$(1 \le t \le T)$ Viterbi sử dụng cấu trúc dữ liệu dạng bảng để lưu lại xác suất mà trạng thái ẩn j xuất hiện sau khi biết được dãy t quan sát và xem xét đến trạng thái ẩn có thể xảy ra nhất từ chuỗi $t-1$ trạng thái xem xét được trước đó. Rõ ràng hơn, thuật toán Viterbi là một dạng thuật toán tìm đường đi tối ưu.

procedure VITERBI_ALGORITHM$(a[1..N,1..N], b[1..N,1..K], \pi[1..N], o[1..T])$
    $viterbi[1..N,1..T] = \{0\}$
    $back\_pointer[1..N,1..T] = \{0\}$
    $viterbi[,1] = \pi * b[,o[1]]$
    for each $t \in (2,T)$ **do**
        for each $j \in (1,N)$ **do**
            $viterbi[j,t] = max_{i=1}^{N} viterbi[i,t-1] * a[i,j] * b[j,o[t]]$
            $back\_pointer[j,t] = argmax_{i=1}^{N} viterbi[i,t-1] * a[i,j] * b[j,o[t]]$
        end for
    end for
    probable_sequence is inferred from *back_pointer*
end procedure

Chọn thao tác cơ sở là $viterbi[j,t] = max_{i=1}^{N} viterbi[i,t-1] * a[i,j] * b[j,o[t]]$ ta có biểu thức biểu diễn độ phức tạp thuật toán như sau:

$$Complexity = O(\sum_{t=2}^{T}\sum_{j=1}^{N} N) \approx O(N^2.T)$$

5 Cho một chuỗi quan sát, giả sử ta cho rằng chuỗi quan sát này được sinh ra từ một mô hình Markov ẩn với tham số chưa biết, thuật toán Baum-Welch được dùng để ước lượng các tham số này. Thuật toán Baum-Welch là trường hợpđặc biệt của thuật toán Kỳ vọng tối ưu (Expectation-maximization, hay EM). Thuật toán này gồm 2 bước: bước E (Expectation, hay kỳ vọng) và bước M (Maximization, hay tối ưu)

5.1 Mô tả thuật toán Kỳ vọng-Tối ưu tổng quát

Thuật toán thường được dùng trong các model thống kê để xác định các tham số sao cho độ hợp lí cục bộ (local likelihood) đạt được là lớn nhất và được dùng trong trường hợp có liên quan đến các biến bị ẩn và dữ liệu bị khuyết.

Thuật toán EM có thể mô tả một cách tổng quát như sau:

1. Xét trên bộ dữ liệu quan sát. Đầu tiên thuật toán EM sẽ gán giá trị khởi điểm cho các tham số.
2. Bước kì vọng (Exceptiom step hay E-step): Sử dụng bộ dữ liệu quan sát được có sẵn để dự đoán ra các giá trị của dữ liệu bị khuyết
3. Bước tối ưu (Maximization step hay M-step): Sử dụng bộ dữ liệu đã điền khuyết ở bước kì vọng (bước 2) để cập nhật lại các tham số cần tìm.
4. Lặp lại bước 2 và bước 3 cho đến khi hội tụ.

5.2 Mô tả và đánh giá độ phức tạp của bước E và bước M của thuật toán Baum-Welch

Mục tiêu của bài toán này là tìm cách cập nhật lại các tham số của ma trận A và B trong mô hình $\lambda = (A, B, \pi)$ sao cho cực đại hoá được xác suất $(O|\pi)$.

Thuật toán Backward

Với bài toán này, ta cần định nghĩa một mảng $\beta_t(i)$ là xác suất trạng thái S_i tại thời điểm t và đã quan sát được đoạn từ O_{t+1} về cuối từ mô hình λ cho trước:

$$\beta_t(i) = P(o_{t+1}, o_{t+2}...o_T | q_t = S_i, \lambda)$$

Dựa trên ý tưởng quy hoạch động giống với thuật toán Forward ta có thể tính được mảng β dễ dàng với thuật toán Backward.

Thuật toán Baum-Welch

Bước Exception

Tại bước này ta định nghĩa $\xi_t(i,j)$ là xác suất ở trạng thái S_i tại thời điểm t và rơi vào trạng thái S_j tại thời điểm $t+1$ với mô hình λ cho trước và chuỗi quan sát O:

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda)$$

Và định nghĩa $\gamma_t(i)$ là xác suất ở trạng thái S_i vào thời điểm t với mô hình λ và chuỗi quan sát O:

$$\gamma_t(i) = P(q_t = S_i | O, \lambda)$$

Lúc này ta dựa vào mảng α và β. Trong đó $\alpha_t(i)$ là xác suất quan sát được trạng thái trạng thái S_i tại thời điểm t và đã quan sát được đoạn $O_{1..t}$ và tượng tự với β là quan sát được đoạn $O_{t+1..T}$.

Khi đó ta có thể tính được $\xi_t(i,j)$ như sau:

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda)$$

$$\Leftrightarrow \xi_t(i,j) = \frac{P(q_t = S_i, q_{t+1} = S_j, O|\lambda)}{P(O|\lambda)}$$

$$\Leftrightarrow \xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)}$$

Tương tự ta tính được $\gamma_t(i)$:

$$\gamma_t(i) = P(q_t = S_i | O, \lambda)$$

$$\Leftrightarrow \gamma_t(i) = \frac{P(q_t = S_i, O|\lambda)}{P(O|\lambda)}$$

$$\Leftrightarrow \gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)}$$

Nếu ta lấy tổng theo $t \in [1, T-1]$ của $\xi_t(i,j)$ khi đó ta nhận được là số lần kỳ vọng chuyển trạng thái từ S_i qua S_j. Tương tự khi lấy tổng theo $t \in [1, T-1]$ của $\gamma_t(i)$, kết quả nhận được sẽ là số lần kỳ vọng ở trạng thái S_i.

Bước Maximization

Với các đại lượng đã tính ở trên, ta có thể cập nhật lại các giá trị cho tham số A và B.

Cụ thể ta có thể ước lượng cho tham số A:

$$\widehat{a}_{ij} = \frac{\text{Số lần kỳ vọng chuyển trạng thái từ } S_i \text{ sang } S_j}{\text{Số lần kỳ vọng chuyển trạng thái từ } S_i}$$

$$\Leftrightarrow \widehat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{k=1}^{N} \xi_t(i,k)}$$

Và ước lượng cho tham số B:

$$\widehat{b}_i(v_k) = \frac{\text{Số lần kỳ vọng ở trạng thái } S_i \text{ quan sát được } v_k}{\text{Số lần kì vọng ở trạng thái } S_i}$$

$$\widehat{b}_i(v_k) = \frac{\sum_{t=1 \wedge O_t = v_k}^{T-1} \gamma_t(i)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

Mã giả

procedure BAUM_WELCH_ALGORITHM(o[1..M, π[1..M], V[1..K])
    Initialize A và B
    $likelihood\_prob \leftarrow FORWARD\_ALGORITHM(A, B, \pi)$
    while not convergence **do**
        Build α and β from current A and B
        for $t \leftarrow 1$ to $T-1$ **do**
            for $i \leftarrow 1$ to N **do**
                for $j \leftarrow 1$ to N **do**
                    $\xi_t(i,j) = \dfrac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{likihood\_prob}$
                end for
            end for
        end for
        for $t \leftarrow 1$ to $T-1$ **do**

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

$\gamma_t(i) = \dfrac{\alpha_t(i)\beta_t(i)}{likelihood\_prob}$

end for

end for

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

for $j \leftarrow 1$ to N **do**

$\widehat{a}_{ij} = \dfrac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{k=1}^{N} \xi_t(i,k)}$

end for

end for

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

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

$\widehat{b}_i(v_k) = \dfrac{\sum_{t=1 \wedge O_t = v_k}^{T-1} \gamma_t(i)}{\sum_{t=1}^{T-1} \gamma_t(i)}$

end for

end for

end while

return (A, B)

end procedure

Đô phức tạp tại bước E: $O(N^2T + NT) \approx O(N^2T)$

Độ phức tạp tại bước M: $O(N^2T + NKT)$