ГЛАВА 1. ОСНОВЫ ТЕОРИИ СКРЫТЫХ МАРКОВСКИХ МОДЕЛЕЙ

В данной главе рассматриваются основные положения теории СММ, применительно к задаче распознавания лиц. Исследуется вопрос формирования наблюдаемых состояний.

1.1. Элементы скрытых марковских моделей

Элементами СММ являются:

1. Конечное множество скрытых состояний $S = \{s_1, s_2, ..., s_N\}$.

2. Конечное множество наблюдаемых состояний (или дискретный алфавит) $V = \{\nu_1, \nu_2, ..., \nu_M\}$.

3. Матрица переходных вероятностей $A = \{a_{ij}\}$, где $a_{ij} = P(q_t = s_j \mid q_{t-1} = s_i)$, $i, j = \overline{1, N}$, $t = \overline{1, T}$, ассоциируемая со стационарной марковской цепью на пространстве скрытых состояний. Здесь q_t – состояние модели в момент времени t.

Для матрицы A выполняется условие нормировки: $0 \le a_{ij} \le 1, \quad i, j = \overline{1, N}$, $\sum_{i=1}^{N} a_{ij} = 1, \quad j = \overline{1, N}$.

Последовательность $Q = q_1, q_2, ..., q_T$ случайных величин со значениями в S удовлетворяет следующему условию:

$$P(q_t = s_{i_t} \mid q_{t-1} = s_{i_{t-1}}, ..., q_1 = s_{i_1}) = P(q_t = s_{i_t} \mid q_{t-1} = s_{i_{t-1}}), \ \forall t, i_1, ..., i_t.$$

Таким образом, марковская цепь является стационарной цепью первого порядка.

4. Начальная вероятность состояний $\pi = \{\pi_i\}$, $\pi_i = P(q_1 = s_i)$, $i = \overline{1, N}$. При этом $\pi_i \ge 0, \quad i = \overline{1, N}$, $\sum_{i=1}^{N} \pi_i = 1$.

5. Матрица вероятностей наблюдаемых символов (матрица эмиссий) $B = \{b_i(k)\}$, $b_i(k) = P(o_t = \nu_k \mid q_t = s_i)$, $i = \overline{1, N}$, $k = \overline{1, M}$, o_t – наблюдаемый символ в

момент времени $t = \overline{1,T}$. Так же выполняется условие нормировки: $b_i(k) \geq 0$, $\sum_{k=1}^{M} b_i(k) = 1, \quad i = \overline{1,N}$.

Каждой последовательности скрытых состояний $s_{i_1}, s_{i_2}, \ldots$ ставится в соответствие последовательность случайных величин $o_1, o_2, \ldots$ со значениями в V. При этом выполняется следующие условие:

$$P(o_t = \nu_{j_t} \mid o_1 = \nu_{j_1}, \ldots, o_{t-1} = \nu_{j_{t-1}}, o_{t+1} = \nu_{j_{t+1}}, \ldots, o_T = \nu_{j_T}, q_1 = s_{i_1}, \ldots, q_t = s_{i_T}) =$$
$$= P(o_t = \nu_{j_t} \mid q_t = s_{i_t}), \quad \forall t, i_1, \ldots, i_T, j_1, \ldots, j_T.$$

То есть вероятность наблюдения некоторого символа зависит только от того, в каком состоянии находится модель в данный момент времени.

1.2. Этап обучения

Входные данные для системы распознавания – это наборы изображений лиц, соответствующие различным людям. Каждой персоне ставится в соответствие СММ, подбор параметров которой происходит в процессе обучения. На этом этапе каждая модель $\lambda_r = (\pi_r, A_r, B_r)$, $r = \overline{1,R}$ (R – число персон) настраивается на свои обучающие изображения в результате максимизации функций $P(O^{(r)}, Q \mid \lambda_r)$ и $P(O^{(r)} \mid \lambda_r)$. Определение максимизирующей последовательности скрытых состояний при фиксированной последовательности наблюдаемых состояний, дает алгоритм динамического программирования Витерби. Алгоритм Баум-Велша используется для максимизации вероятности наблюдения последовательности $O^{(r)}$ при использовании модели λ_r. В результате обучения полученная модель λ_r может быть использована в качестве генератора наблюдаемой последовательности $O^{(r)}$ (извлеченных из обучающих изображений), а также при распознавании сигналов (тестовых изображений) в некотором смысле близких к $O^{(r)}$.

В [15] с выводом многих формул и примерами приведены все шаги обучения. Для ознакомления с проблемой обучения СММ так же подходят [7] и [18].

Задача обучения состоит в том, чтобы подобрать параметры модели λ так, чтобы она правильно распознавала исходные данные $O = O^1, O^2, ..., O^K$. Необходимо выбрать один из способов обучения [49]:

1) распознать эти последовательности наблюдений, сравнить результаты распознавания $Q^{*1}, Q^{*2}, ..., Q^{*K}$ с правильными ответами $Q^1, Q^2, ..., Q^K$, вычислить в каком либо смысле среднюю ошибку и минимизировать ее, варьируя λ;

2) распознать последовательность мультинаблюдений и максимизировать функцию правдоподобия наблюдения последовательности O, в предположении, что последовательность скрытых состояний найдена правильно. То есть максимизировать $\prod_{k=1}^{K} P(O^k \mid Q^{*k})$, варьируя λ;

3) максимизировать функцию правдоподобия от наблюдений, то есть максимизировать $P(O) = \prod_{k=1}^{K} P(O^k)$, варьируя λ.

Заметим, что в качестве оптимизационного критерия можно использовать не только максимум правдоподобия: $P_r^* = \max_{\lambda_r} P(O^r \mid \lambda_r)$, но и максимум взаимной информации: $I_r^* = \max_{\lambda_r} \left[\log P(O^r \mid \lambda_r) - \log \sum_{w=1}^{R} P(O^w \mid \lambda_w) \right]$ или минимизировать различающую информацию [26].

Первый способ обучения – это обучение с учителем. Его можно проводить, например, методом градиентного спуска, и он всем хорош, кроме своей трудоемкости. Остальные два способа – это обучение без учителя, хотя во втором способе можно использовать учителя. Чаще всего применяется третий способ обучения (иногда с последующим доучиванием другими способами), поскольку для него известен быстрый алгоритм.

Этот алгоритм, в общей ситуации называемый ОМ (максимизация ожидания; EM – expectation maximization) или, конкретно для скрытых марковских моделей, алгоритмом Баума-Велша. Данный алгоритм является итеративным и сходится, вообще говоря, не к глобальному максимуму правдоподобия, а к локальному. Кроме этого метода решения возможно так же использовать и другие методы оптимизации для поиска максимума функции правдоподобия. Однако, в [7] отмечено, что алгоритм Гаусса-Ньютона имеет тенденцию к несходимости и зависит от хорошего начального выбора параметров модели λ. Там же кроме алгоритма Баум-Велша рассмотрен метод градиентного спуска Болди-Чавина. В [32] проводится аналогия между ОМ алгоритмом и градиентными методами.

Вероятность того, что последовательность $Q = q_1, q_2, ..., q_T$ скрытых состояний порождает последовательность $O = o_1, o_2, ..., o_T$ наблюдений равна

$$P(O \mid Q, \lambda) = P(o_1, o_2, ..., o_T \mid q_1, q_2, ..., q_T) = \prod_{t=1}^{T} P(o_t \mid q_t, \lambda) = \prod_{t=1}^{T} b_{q_t}(o_t).$$

Вероятность появления последовательности $q_1, q_2, ..., q_T$ вычисляется как:

$$P(Q \mid \lambda) = P(q_1, q_2, ..., q_T) = \pi_{q_1} \prod_{t=1}^{T-1} a_{q_t q_{t+1}}.$$

Обе эти формулы следуют непосредственно из условий независимости в определении скрытой марковской модели. Вероятность наблюдения последовательности O, порожденной последовательностью Q, равна

$$P(O, Q \mid \lambda) = P(O \mid Q, \lambda) P(Q, \lambda) = \pi_{q_1} \prod_{t=1}^{T-1} a_{q_t q_{t+1}} \prod_{t=1}^{T} b_{q_t}(o_t) \qquad (1.1)$$

и вычисляется за $2T - 1$ умножений. Вероятность наблюдения последовательности O без каких-либо условий по определению равна

$$P(O \mid \lambda) = \sum_{q_1, q_2, \ldots, q_T} P(O, Q \mid \lambda) = \sum_{q_1, q_2, \ldots, q_T} \left(\pi_{q_1} \prod_{t=1}^{T-1} a_{q_t q_{t+1}} \prod_{t=1}^{T} b_{q_t}(o_t) \right). \quad (1.2)$$

Вычислительная сложность (1.1) порядка $2TN^T$ операций.

1.2.1. Алгоритм прямого-обратного прохода

Согласно [26] для эффективного вычисления вероятности (1.1) используют алгоритм прямого-обратного прохода (Forward-Backward). Данный алгоритм основывается на методе динамического программирования.

Пусть *forward-вероятность:*

$$\alpha_t(i) = P(o_1, o_2, \ldots, o_t, q_t = s_i \mid \lambda) = \sum_{q_1, q_2, \ldots, q_t \mid q_t = s_i} P(o_1, o_2, \ldots, o_t, q_1, q_2, \ldots, q_t \mid \lambda),$$

то есть вероятность того, что данная последовательность символов $o_1, o_2, \ldots, o_t$ будет сгенерирована моделью λ и эта модель находится в состоянии s_i.

Для вычисления α необходимо провести следующие шаги.

1. Инициализация:

$$\alpha_1(i) = \pi_i b_i(o_1), \quad i = \overline{1, N}.$$

2. Индукция:

$$\alpha_{t+1}(i) = b_i(o_{t+1}) \left[\sum_{j=1}^{N} \alpha_t(j) a_{ji} \right], \quad i = \overline{1, N}, \quad t = \overline{1, T-1}. \quad (1.3)$$

3. Завершение:

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i). \quad (1.4)$$

Таким способом $P(O|\lambda)$, а значит и все $\alpha_t(i)$ при $i=\overline{1,N}$, $t=\overline{1,T}$, можно вычислить за порядка $3Tn^2$ операций. Более того [49], множитель n^2 – это количество элементов матрицы перехода. Если матрица перехода разреженная и в ней только n' элементов могут отличаться от нуля, то вероятность (1.4) может быть вычислена соответственно за $3Tn'$ операций.

Формула (1.3) может быть получена следующим образом:

$$\alpha_{t+1}(i)=P(o_1,o_2,...,o_{t+1},q_{t+1}=s_i|\lambda)=\sum_{j=1}^{N}P(o_1,o_2,...,o_t,o_{t+1},q_t=s_j,q_{t+1}=s_i|\lambda)=$$

$$=\sum_{j=1}^{N}P(o_1,o_2,...,o_t,q_t=s_j,q_{t+1}=s_i|\lambda)P(o_{t+1}|q_{t+1}=s_i)=$$

$$=b_i(o_{t+1})\sum_{j=1}^{N}P(o_1,o_2,...,o_t,q_t=s_j|\lambda)P(q_{t+1}=s_i|q_t=s_j)=b_i(o_{t+1})\left[\sum_{j=1}^{N}\alpha_t(j)a_{ji}\right].$$

Backward-вероятность:

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

то есть это вероятность того, что последовательность наблюдений $o_{t+1},o_{t+2},...,o_T$, данная состоянием s_i, будет сгенерирована моделью λ.

Для вычисления β необходимо провести следующие шаги.

1. Инициализация:

$$\beta_T(i)=1,\quad i=\overline{1,N}.$$

2. Индукция:

$$\beta_t(i)=\sum_{j=1}^{N}\beta_{t+1}(j)b_j(o_{t+1})a_{ij},\quad i=\overline{1,N},\quad t=\overline{1,T-1}. \tag{1.5}$$

3. Завершение:

$$P(O \mid \lambda) = \sum_{i=1}^{N} \beta_1(i). \tag{1.6}$$

Вычислительные сложности (1.5) и (1.6) эквивалентны.

Вывод формулы (1.5):

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, ..., o_T \mid q_t = s_i, \lambda) = \sum_{j=1}^{N} P(o_{t+1}, o_{t+2}, ..., o_T, q_{t+1} = s_j \mid q_t = s_i, \lambda) =$$

$$= \sum_{j=1}^{N} P(o_{t+2}, ..., o_T \mid q_t = s_i, \lambda) P(o_{t+1} \mid q_t = s_i) P(q_{t+1} = s_j \mid q_t = s_i) = \sum_{j=1}^{N} \beta_{t+1}(j) b_i(o_{t+1}) a_{ij}.$$

Кроме того, заметим [19], что $P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_t(i)\beta_t(i) \;\; \forall t = \overline{1,T}$.

В [19], [26] приводятся иллюстрация, поясняющая данный алгоритм (см. рисуноки 1.1).

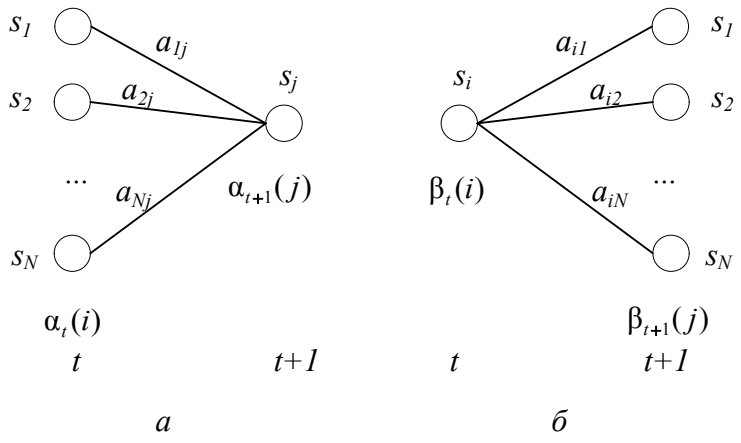

Рисунок 1.1. Иллюстрация к алгоритму Forward-Backward

На рисунке 1.1, *а* показано, как состояние s_j в момент времени $t+1$ может быть достигнуто из N возможных состояний в момент времени t. На рисунке 1.1, *б* показано, как состояния s_i, $i = \overline{1,N}$ в момент времени $t+1$ могут быть достигнуты из состояния s_j в момент времени t.

1.2.2. Алгоритм Баум-Велша

На этапе обучения СММ необходимо произвести настройку параметров модели $\lambda = (A, B, \pi)$ в результате решения задачи [26]: $P(O \mid \lambda) \to \max$. Для решения данной задачи, как мы уже отмечали, можно использовать метод Баум-Велша. В [4], [7], [13], [15], [21], [49] достаточно подробно приведен вывод формул для этого метода. Фактически необходимо подобрать последовательность скрытых состояний Q к последовательности наблюдений O [52], то есть решить задачу с недостающими (пропущенными) данными. Эту проблему решают при помощи ОМ-алгоритма, который ориентирован на поиск максимума функции правдоподобия по параметрам ненаблюдаемой функции распределения для множества наблюдений, где данные не полны или имеются пропуски. Для модели смесей общая схема ОМ алгоритма приведена в [4], [34]. В такой модели недостающие данные – это переменные, указывающие, из какого компонента смеси извлечен элемент данных (см. формулу (2.2)).

Алгоритм Баум-Велша.

Зафиксируем обучающий набор O из K последовательностей наблюдений O^k длин T^k, $k = \overline{1, K}$ и начальный набор параметров модели λ^0. Будем пытаться увеличить $P(O \mid \lambda)$ или, что то же самое, уменьшить $E(O \mid \lambda) = -\ln P(O \mid \lambda)$. Из формулы (1.2) следует, что:

$$P(O \mid \lambda) = \prod_{k=1}^{K} P(O^k \mid \lambda) = \prod_{k=1}^{K} \sum_{q_1, q_2, \ldots, q_{T^k}} P(O^k, Q \mid \lambda) = \prod_{k=1}^{K} \sum_{q_1, q_2, \ldots, q_{T^k}} \left(\pi_{q_1} \prod_{t=1}^{T^k - 1} a_{q_t q_{t+1}} \prod_{t=1}^{T^k} b_{q_t}(o_t^k) \right).$$

На каждом шаге итерации функция ошибки мажорируется функцией более простого вида, минимум которой единственен и находится явно:

$$E(O,\lambda)-E(O,\lambda^0)=-\sum_{k=1}^{K}\ln\frac{P(O^k\mid\lambda)}{P(O^k\mid\lambda^0)}=-\sum_{k=1}^{K}\ln\frac{\sum_{q_1,q_2,\ldots,q_{T^k}}P(O^k,Q\mid\lambda)}{P(O^k\mid\lambda^0)}=$$

$$=-\sum_{k=1}^{K}\ln\sum_{q_1,q_2,\ldots,q_{T^k}}\frac{P(O^k,Q\mid\lambda)}{P(O^k\mid\lambda^0)}=-\sum_{k=1}^{K}\ln\sum_{q_1,q_2,\ldots,q_{T^k}}\frac{P(Q\mid O^k,\lambda^0)}{P(Q\mid O^k,\lambda^0)}\frac{P(O^k,Q\mid\lambda)}{P(O^k\mid\lambda^0)}=$$

$$=-\sum_{k=1}^{K}\ln\sum_{q_1,q_2,\ldots,q_{T^k}}P(Q\mid O^k,\lambda^0)\frac{P(O^k,Q\mid\lambda)}{P(O^k,Q\mid\lambda^0)}\le$$

$$\le-\sum_{k=1}^{K}\sum_{q_1,q_2,\ldots,q_{T^k}}P(Q\mid O^k,\lambda^0)\ln\frac{P(O^k,Q\mid\lambda)}{P(O^k,Q\mid\lambda^0)}=$$

$$=-\sum_{k=1}^{K}\sum_{q_1,q_2,\ldots,q_{T^k}}P(Q\mid O^k,\lambda^0)\ln P(O^k,Q\mid\lambda)+\sum_{k=1}^{K}\sum_{q_1,q_2,\ldots,q_{T^k}}P(Q\mid O^k,\lambda^0)\ln P(O^k,Q\mid\lambda^0).$$

Неравенство в этой цепочке преобразований следует из выпуклости функции $-\ln(\cdot)$ и того, что условные вероятности неотрицательны и их сумма по всем возможным последовательностям Q длины T^k равна 1.

Обозначим $Q(O,\lambda^0,\lambda)=-\sum_{k=1}^{K}\sum_{q_1,q_2,\ldots,q_{T^k}}P(Q\mid O^k,\lambda^0)\ln P(O^k,Q\mid\lambda)$. Тогда полученное неравенство означает, что $E(O,\lambda)\le Q(O,\lambda^0,\lambda)-Q(O,\lambda^0,\lambda^0)+E(O,\lambda^0)$, то есть правая часть мажорирует $E(O,\lambda)$ и совпадает с ней в точке λ^0. Значит, если удастся найти точку λ глобального минимума правой части или, что то же самое, точку минимума $Q(O,\lambda^0,\cdot)$ по всей области определения, то $E(O,\lambda)\le E(O,\lambda^0)$. Прямые вычисления производных $Q(O,\lambda^0,\lambda)$ по параметрам, показывают, что критическая точка $Q(O,\lambda^0,\lambda)$ всегда единственна, и является точкой минимума.

Далее определим [4]:

$$\gamma_t(i)=P(q_t=s_i\mid O,\lambda)=\frac{\alpha_t(i)\beta_t(i)}{P(O\mid\lambda)},\quad i=\overline{1,N},\quad t=\overline{1,T-1}. \tag{1.7}$$

Равенство (1.7) следует из того факта, что

$$\alpha_t(i)\beta_t(i) = P(o_1, o_2, ..., o_t, q_t = s_i \mid \lambda) P(o_{t+1}, o_{t+2}, ..., o_T \mid q_t = s_i, \lambda) = P(O, q_t = s_i \mid \lambda).$$

Введем следующую переменную:

$$\xi_t(i, j) = P(q_t = s_i, q_{t+1} = s_j \mid O, \lambda) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{P(O \mid \lambda)}, i, j = \overline{1, N}, \quad t = \overline{1, T-1}. \qquad (1.8)$$

Формула (1.8) следует из того, что:

$$\xi_t(i, j) = P(q_t = s_i, q_{t+1} = s_j \mid O, \lambda) = \frac{P(q_t = s_i, q_{t+1} = s_j, O \mid \lambda)}{P(O \mid \lambda)} =$$

$$= \frac{P(o_1, ..., o_{t+1}, q_t = s_i, q_{t+1} = s_j \mid \lambda) P(o_{t+2}, ..., o_T \mid q_{t+1} = s_j, \lambda)}{P(O \mid \lambda)} =$$

$$= \frac{P(o_1, ..., o_{t+1}, q_t = s_i \mid \lambda) P(o_{t+2}, ..., o_T \mid q_{t+1} = s_j, \lambda) P(o_{t+1} \mid q_{t+1} = s_j) P(q_{t+1} = s_j \mid q_t = s_i)}{P(O \mid \lambda)} =$$

$$= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{P(O \mid \lambda)}.$$

На рисунке 1.2 поясняется, каким образом вычисляется совместная вероятность пребывания модели в момент времени t в состоянии s_i и в момент времени $t+1$ – в состоянии s_j.

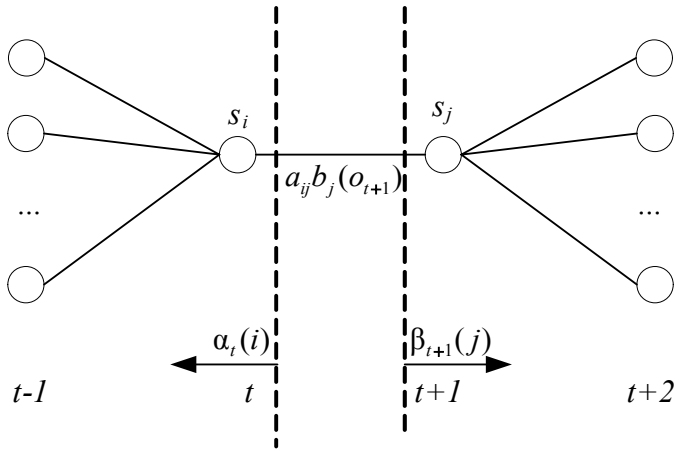

Рисунок 1.2. Связь соседних состояний s_i и s_j

Заметим, что $\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j)$. Введем $\gamma_t(i,m) = P(q_t = i, \omega_{it} = m \mid O, \lambda)$, ω_{it} – случайная переменная, показывающая компоненту смеси в момент времени t для состояния i. Тогда: $\gamma_t(i,m) = \gamma_t(i)\left[\frac{c_{im} N(o_t, \mu_{im}, \Sigma_{im})}{\sum_{m=1}^{M_i} c_{i\tilde{m}} N(o_t, \mu_{i\tilde{m}}, \Sigma_{i\tilde{m}})}\right]$.

Для СММ с непрерывной вероятностью описания наблюдаемых состояний (см. пункт 2.1) минимум функции $Q(O, \lambda^0, \lambda)$ достигается в точке λ^* с координатами (для мультинаблюдения [15], [19]):

$$\pi_i^* = \frac{1}{K}\sum_{k=1}^{K} \gamma_1^{(k)}(i) = \frac{1}{K}\sum_{k=1}^{K} \frac{\alpha_1(i)\beta_1(i)}{P(O^{(k)} \mid \lambda)}, \tag{1.9}$$

$$a_{ij}^* = \frac{\sum_{k=1}^{K}\sum_{t=1}^{T^k-1} \xi_t^{(k)}(i,j)}{\sum_{k=1}^{K}\sum_{t=1}^{T^k-1} \gamma_t^{(k)}(i)} = \frac{\sum_{k=1}^{K} \frac{1}{P(O^{(k)} \mid \lambda)} \sum_{t=1}^{T^k-1} \alpha_t^{(k)}(i) a_{ij} b_j(o_{t+1}^{(k)}) \beta_{t+1}^{(k)}(j)}{\sum_{k=1}^{K} \frac{1}{P(O^{(k)} \mid \lambda)} \sum_{t=1}^{T^k-1} \alpha_t^{(k)}(i)\beta_t^{(k)}(i)}, \tag{1.10}$$

$$c_{im}^* = \frac{\sum_{k=1}^{K}\sum_{t=1}^{T^k-1} \gamma_t^{(k)}(i,m)}{\sum_{k=1}^{K}\sum_{t=1}^{T^k-1} \gamma_t^{(k)}(i)}, \tag{1.11}$$

$$\mu_{im}^* = \frac{\sum_{k=1}^{K}\sum_{t=1}^{T^k-1} \gamma_t^{(k)}(i,m) o_t^{(k)}}{\sum_{k=1}^{K}\sum_{t=1}^{T^k-1} \gamma_t^{(k)}(i,m)}, \tag{1.12}$$

$$\Sigma_{im}^* = \frac{\sum_{k=1}^{K}\sum_{t=1}^{T^k-1} \gamma_t^{(k)}(i,m)(o_t^{(k)} - \mu_{im}^*)(o_t^{(k)} - \mu_{im}^*)^T}{\sum_{k=1}^{K}\sum_{t=1}^{T^k-1} \gamma_t^{(k)}(i,m)}. \tag{1.13}$$

Аналогично (1.9), (1.10) для мультинаблюдений возможно выразить через прямые – обратные переменные и (1.11)-(1.13).

Непосредственно алгоритм Баум-Велша состоит из 3 основных этапов.

1 этап. Forward-Backward алгоритм.

2 этап. Перевычисление параметров модели λ (1.9)-(1.13).

3 этап. Пока не достигнут порог сходимости: $|P(O|\lambda)^{iter-1} - P(O|\lambda)^{iter}| > \varepsilon$, повторять 1-ый и 2-ой этапы.

Начальные значения параметров A и π модели можно задавать произвольно, учитывая вероятностные нормировки. Алгоритм обучения всегда сходится и при этом почти всегда – к точке локального максимума правдоподобия $P(O|\lambda)$. Однако остаются вопросы, на которые нет универсального ответа: с какой скоростью сходится обучение и всегда ли такое обучение обеспечивает хорошее распознавание.

1.2.3. Масштабирование

Во всех вышеописанных вычислениях вероятности умножаются друг на друга, то есть числа не превышающие 1 и имеющие типичные значения обратные количеству состояний (скрытых или наблюдаемых), в количестве, пропорциональном длине последовательности. Для длинных (длины порядка 100 и более) последовательностей эти произведения меньше минимальных аппаратно реализуемых чисел типичных компьютеров. То есть нужно либо программно реализовывать неограниченную точность вычислений, что связано с временными затратами, либо как-то масштабировать все промежуточные результаты, чтобы они не стремились к нулю. Исключением является алгоритм Витерби, в котором вероятности только перемножаются, но никогда не складываются, и можно вместо умножения вероятностей выполнять сложение их логарифмов. Но при обучении масштабирование необходимо. Методы масштабирования, почти не замедляющие обучение, известны [26], [49].

В [15], [18], [26] даны идеи использования параметра масштабирования. Для *forward-вероятности* используют следующие преобразования [15]:

1. Инициализация:

$$\widetilde{\alpha}_1(i) = \alpha_1(i), \quad i = \overline{1, N}.$$

2. Индукция:

$$\widetilde{\alpha}_{t+1}(i) = b_i(o_{t+1})\left[\sum_{j=1}^{N} \alpha'_t(j) a_{ji}\right], \quad i = \overline{1, N}, \quad t = \overline{1, T-1}.$$

Где $\alpha'_t(i) = \dfrac{\sum_{j=1}^{N} \alpha'_{t-1}(j) a_{ji} b_i(o_t)}{\sum_{n=1}^{N}\sum_{j=1}^{N} \alpha'_{t-1}(j) a_{jn} b_n(o_t)}$. Определим параметр масштаба как:

$c_t = \dfrac{1}{\sum_{i=1}^{N} \alpha_t(i)}, \quad t = \overline{1, T}$.

Тогда:

$$\alpha'_t(i) = c_t \widetilde{\alpha}_t(i), \quad i = \overline{1, N}.$$

$$\alpha'_{t-1}(j) = \left(\prod_{\tau=1}^{t-1} c_\tau\right)\alpha_{t-1}(j), \quad j = \overline{1, N}, \quad t = \overline{2, T}.$$

Доказательство приведено в [15].

Для *backward -вероятности* используют аналогичные преобразования:

1. Инициализация:

$$\widetilde{\beta}_T(i) = \beta_T(i), \quad i = \overline{1, N}.$$

2. Индукция:

$$\beta'_t(i) = \sum_{j=1}^{N} \widetilde{\beta}_t(j) a_{ji} b_j(o_{t+1}), \quad i = \overline{1, N}, \quad t = \overline{1, T-1}.$$

Где $\beta'_t(i) = \left(\prod_{\tau=t+1}^{T} c_\tau\right)\beta_t(i)$. Доказательство так же приведено в [15].

Заметим, что $\left(\prod_{\tau=1}^{T} c_\tau\right) P(O \mid \lambda) = 1$, так как:

$$\left(\prod_{\tau=1}^{T} c_\tau\right) P(O \mid \lambda) = \prod_{\tau=1}^{T} c_\tau \left(\sum_{i=1}^{N} \alpha_T(i)\right) = \prod_{\tau=1}^{T-1} c_\tau \left(c_T \sum_{i=1}^{N} \alpha_T(i)\right) = \prod_{\tau=1}^{T-1} c_\tau \left(c_T \sum_{i=1}^{N} \alpha_T(i)\right) =$$

$$= \prod_{\tau=1}^{T-1} c_\tau \frac{\sum_{i=1}^{N} \alpha_T(i)}{\sum_{i=1}^{N} \tilde{\alpha}_T(i)} = \prod_{\tau=1}^{T-1} c_\tau \frac{\sum_{i=1}^{N} \alpha_T(i)}{\sum_{i=1}^{N} b_i(o_T)\left[\sum_{j=1}^{N} \alpha'_{T-1}(j) a_{ji}\right]} =$$

$$= \prod_{\tau=1}^{T-1} c_\tau \frac{\sum_{i=1}^{N} \alpha_T(i)}{\sum_{i=1}^{N} b_i(o_T)\left[\sum_{j=1}^{N} \left(\left(\prod_{\tau=1}^{T-1} c_\tau\right) \alpha_{T-1}(j)\right) a_{ji}\right]} = \frac{\sum_{i=1}^{N} \alpha_T(i)}{\sum_{i=1}^{N} b_i(o_T)\left[\sum_{j=1}^{N} \alpha_{T-1}(j) a_{ji}\right]} =$$

$$= \frac{\sum_{i=1}^{N} \alpha_T(i)}{\sum_{i=1}^{N} \alpha_T(i)} = 1.$$

Далее рассмотрим пересчет параметров модели λ с учетом масштабирования (представим $P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_t(i)\beta_t(i)$).

$$\gamma'_t(i) = \frac{\alpha'_t(i)\beta'_t(i)}{P(O \mid \lambda)'} = \frac{\alpha'_t(i)\beta'_t(i)}{\sum_{i=1}^{N} \alpha'_t(i)\beta'_t(i)} = \frac{\left(\prod_{\tau=1}^{t} c_\tau\right)\alpha_t(i)\left(\prod_{\tau=t+1}^{T} c_\tau\right)\beta_t(i)}{\sum_{i=1}^{N}\left(\prod_{\tau=1}^{t} c_\tau\right)\alpha_t(i)\left(\prod_{\tau=t+1}^{T} c_\tau\right)\beta_t(i)} =$$

$$= \frac{\left(\prod_{\tau=1}^{T} c_\tau\right)\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N}\left(\prod_{\tau=1}^{T} c_\tau\right)\alpha_t(i)\beta_t(i)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}.$$

Кроме того:

$$\xi'_t(i,j)=\frac{\alpha'_t(i)a_{ij}b_j(o_{t+1})\beta'_{t+1}(j)}{P(O\mid\lambda)'}=\frac{\alpha'_t(i)a_{ij}b_j(o_{t+1})\beta'_{t+1}(j)}{\sum_{i=1}^{N}\alpha'_t(i)\beta'_t(i)}=$$

$$=\frac{\left(\prod_{\tau=1}^{t}c_\tau\right)\alpha_t(i)a_{ij}b_j(o_{t+1})\left(\prod_{\tau=t+2}^{T}c_\tau\right)\beta_{t+1}(i)}{\sum_{i=1}^{N}\left(\prod_{\tau=1}^{T}c_\tau\right)\alpha_t(i)\beta_t(i)}=\frac{c_{t+1}\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(i)}{\sum_{i=1}^{N}\alpha_t(i)\beta_t(i)}.$$

С учетом этого формулы для вычисления $a_{im},c_{im},\mu_{im},\Sigma_{im}$ (1.10)-(1.13) изменяются. Например, (1.10) преобразуется в:

$$a'^{*}_{ij}=\frac{\sum_{k=1}^{K}\sum_{t=1}^{T^k-1}\xi'^{(k)}_t(i,j)}{\sum_{k=1}^{K}\sum_{t=1}^{T^k-1}\gamma'^{(k)}_t(i)}=\frac{\sum_{k=1}^{K}\frac{1}{P(O^{(k)}\mid\lambda)}\sum_{t=1}^{T^k-1}c_{t+1}\alpha^{(k)}_t(i)a_{ij}b_j(o^{(k)}_{t+1})\beta^{(k)}_{t+1}(j)}{\sum_{k=1}^{K}\frac{1}{P(O^{(k)}\mid\lambda)}\sum_{t=1}^{T^k-1}\alpha^{(k)}_t(i)\beta^{(k)}_t(i)}.$$

1.2.4. Выбор наблюдаемых состояний

Наблюдаемые состояния V выбираются как некоторые геометрические классы в пространстве наблюдений O: необходимо разделить наблюдения $o_1,o_2,...,o_T$ по группам. То есть кластеризивать T обучающих векторов на множество M кодовых векторов. Для этого мы использовали 2 алгоритма: кластеризацию k-средних и байесовский классификатор.

1.2.4.1. Алгоритм k-средних

В этом методе задается число k ($k=M$) желаемых кластеров. На первом этапе все наблюдения случайным образом распределяются по кластерам. В дальнейшем алгоритм изменяет принадлежность наблюдений к кластерам таким образом, чтобы, с одной стороны, уменьшить изменчивость наблюдений внутри кластеров (например, минимизировать сумму квадратов отклонений от среднего значения кластера), а с другой – максимизировать изменчивость между кластерами. К преимуществам метода можно отнести его простоту и высокую скорость работы. В ка-

честве недостатков можно отметить необходимость задавать число кластеров, которое может быть неизвестно, а также то, что он способен выделять только кластеры не слишком сложной формы.

Согласно [2] задача этого блока – минимизировать функцию $f = \sum_{a=1}^{M} f_a$, где функция $f_a = \sum_{x \in G[a]} \|x - c_a\|^2$ есть сумма квадратов расстояний от наблюдения x (то есть наблюдаемого символа), принадлежащего данному кластеру $G[a]$ до центра масс (среднего) этого кластера. В качестве нормы используем норму из пространства L_2: $\|z\|^2 = \sum_{i=1}^{\dim(z)} z_i^2$.

Схема выполнения кластеризации.

1. Пусть G_a – текущий кластер x;
2. Определяем:

$$\Delta_a^- = J(G_a) - J(G_a \setminus \{x\}) = \|x - c_a\|^2.$$

3. Среди всех $b = \overline{1, N}, \quad b \neq a$ находим наименьшее

$$\Delta_b^+ = J(G_b + \{x\}) - J(G_b) = \|x - c_b\|^2.$$

4. Если $\Delta_b^+ < \Delta_a^-$, то наблюдение x перемещаем в кластер G_b из G_a.

Там же [2] приводится современная модификация этого алгоритма.

1. Находим $a = \arg\min \|x - c_a\|$.
2. Наблюдение x переносим в кластер G_a .
3. Пересчитываем центры классов c_a.

1.2.4.2. Байесовский классификатор

Наиболее важным свойством, используемым при разделении наблюдений на кластеры, является *плотность распределения* объектов внутри данных кластеров

[34], [36], [51]. Это свойство дает нам возможность определить кластер в виде скопления точек в многомерном пространстве, относительно более плотного по сравнению с иными областями этого пространства, которые либо вообще не содержат точек, либо содержат малое количество наблюдений. Байесовский подход основан на предположении, что плотности распределения классов либо известны, либо могут быть оценены по обучающей выборке. Это очень сильное предположение. Оно позволяет выписать искомый алгоритм в явном аналитическом виде. Более того, можно доказать, что этот алгоритм является оптимальным, обладая минимальной вероятностью ошибки классификации.

Каноническая форма классификатора задается с помощью разделяющих функций, а именно $x \in \omega_i$, $g_i(x) > g_j(x)$, $\forall j \neq i$, $i, j = \overline{1, M}$. Здесь ω_i – ситуация на входе классификатора ω, x – наблюдение, M – число классов. В качестве решающей функцию $g_i(x) = P(\omega_i \mid x)$ можно использовать непосредственно ее или любую монотонную функцию $f(g_i(x))$, если это упрощает вычисления. Будем использовать решающую функцию $g_i(x) = \ln P(x \mid \omega_i) + \ln P(\omega_i)$.

Поскольку распределение наблюдения, по состояниям V подчиняется нормальному закону распределения: $P(x \mid \omega_i) = N(\mu_i, \Sigma_i)$, то

$$g_i(x) = -\frac{1}{2}(x - \mu_i)^T \Sigma_i^{-1}(x - \mu_i) - \frac{l}{2}\ln 2\pi - \frac{1}{2}\ln\left|\Sigma_i\right| + \ln P(\omega_i).$$

Проведенные нами исследования показали, что:

1. Алгоритма *k*-средних чувствителен к начальной сегментации наблюдений по наблюдаемым состояниям. Это влияет не только на скорость сходимости, но и в итоге на процент распознавания.

2. Использование алгоритма *k*-средних на начальной итерации и его современной модификации на остальных (в процессе обучения) также не дает приемлемой сходимости.

3. Экспериментальным путем мы пришли к следующей схеме:

Деление на классы наблюдений, принадлежащих одному скрытому состоянию происходит регулярным образом, а не случайным. Далее выполняется обычный алгоритм кластеризации *k*-средних.

Какого-либо существенного преимущества выбора алгоритма *k*-средних перед байесовским классификатором обнаружено не было.

1.2.5. Обучение СММ

Алгоритм Баум-Велша требует начальных приближений параметров модели λ. Для этого в [26] предлагается использовать так называемый в области распознавания речи *обучающий алгоритм сегментации*. В [26] (см. так же [29]) использовался сегментация k-средних.

Дале приведены этапы обучения СММ.

1 этап. Инициализация модели: задание последовательности скрытых состояний и выбор наблюдаемых состояний.

2 этап. Кластеризация наблюдаемых состояний: для модели *PB* – кластеризация наблюдений внутри каждого скрытого состояния по M_i кластерам. Для модели *GB* – данный шаг отсутствует.

3 этап. Определение наилучшей последовательности скрытых состояний, используя алгоритм Витерби (см. пункт 1.3).

4 этап. Перевычисление параметров с использованием следующих соотношений:

$$c_{im} = \frac{\sum_{t=1}^{T} \delta_{im}(t)}{\delta_i},$$

$$\mu_{im} = \frac{\sum_{t=1}^{T} o_t \delta_{im}(t)}{\delta_i}, \tag{1.14}$$

$$\Sigma_{im} = \frac{\sum_{t=1}^{T} (o_t - \mu_{im})(o_t - \mu_{im})^T \delta_{im}(t)}{\delta_i - 1}. \tag{1.15}$$

Здесь $\delta_i = \sum_{t=1}^{T}\sum_{\tilde{m}=1}^{M_i}\delta_{i\tilde{m}}(t)$, $\delta_{i\tilde{m}}(t)$ – индикаторная переменная, принимающая значение 1 или 0 в зависимости от принадлежности наблюдения o_t i-ому скрытому состоянию и $\tilde{m}$-ому наблюдаемому состоянию.

5 этап. Пока не достигнут порог сходимости, т.е. $\left|P(O,Q^{iter}\,|\,\lambda) - P(O,Q^{iter-1}\,|\,\lambda)\right| > \varepsilon$ повторять 2 этап-5 этап.

На рисунке 1.3 приведены значения $\ln b_i(o_t)$ в зависимости от номера наблюдения, полученные в результате обучения на этапах 1-5.

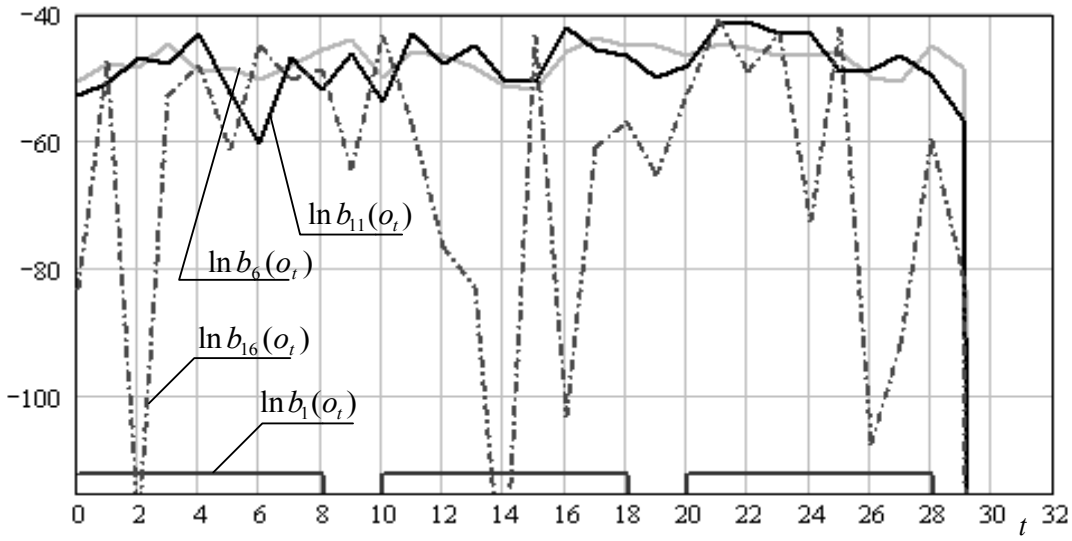

Рисунок 1.3. Значения $\ln b_i(o_t)$ в зависимости от номера наблюдения

Для всех остальных состояний s_i , $i = \overline{1,25}$, $i \neq \{1,6,11,16\}$ вероятность наблюдения символа o_t равна $b_i(o_t) = 0$, $t = \overline{1,32}$. Полученная последовательность скрытых состояний совпадает с последовательностью состояний для которых в момент времени t значение $\ln b_i(o_t)$ максимально.

6 этап. Алгоритм Баум-Велша.

Заметим, что (1.14)-(1.15) есть оценки максимального правдоподобия для параметров нормального распределения.

На рисунке 1.4 приведены значения логарифма $P(O|\lambda)$ в зависимости от номера итерации для разных персон (s1-s5). Скачок в значении вероятности происходит при начале работы алгоритма Баум-Велша.

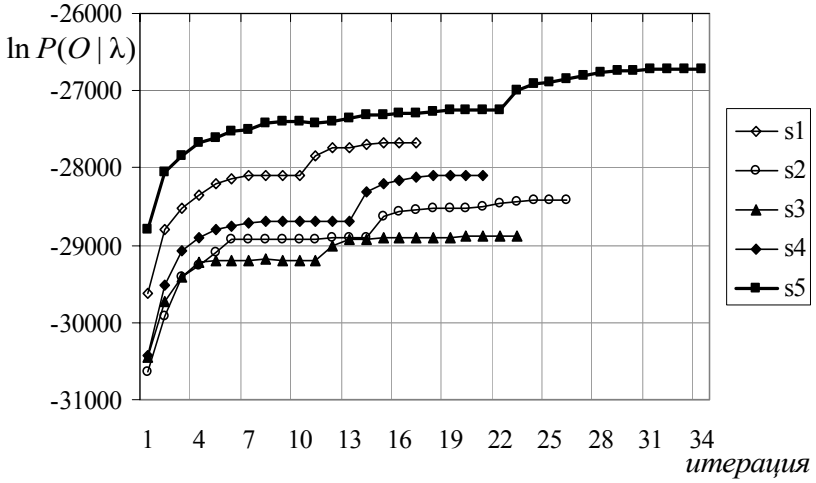

Рисунок 1.4. Зависимость вероятности $P(O|\lambda)$ от номера итерации для разных персон (s1-s5)

1.3. Этап распознавания

Задача распознавания состоит в том, чтобы по имеющейся последовательности наблюдений $O = o_1, o_2, ..., o_T$ восстановить последовательность скрытых состояний $Q = q_1, q_2, ..., q_T$, порождающую эти наблюдения с наибольшей вероятностью и/или найти саму эту вероятность. Искать максимум перебором – неприемлемо долго ($O(TN^T)$ вычислений формулы (1.2)). Однако можно воспользоваться методом динамического программирования Витерби. Выигрыш в скорости получается такой же, как при использовании формулы (1.4) – всего порядка $O(N^2T)$ операций.

Заметим, что результат поиска оптимальной последовательности состояний может быть не единственен [26]. Это обусловлено существованием нескольких приемлемых критериев оптимальности. Например, один из возможных критериев состоит в том, чтобы выбрать такие состояния q_t, каждое из которых, взятое отдельно, является наиболее вероятным. Можно также, максимизировать вероятность появления пар состояний (q_t, q_{t+1}) или троек состояний (q_t, q_{t+1}, q_{t+2}) и т. д. В [48] предлагается максимизировать длительность пребывания системы в текущем состоянии.

Однако наиболее широко используется критерий, основанный на поиске *единственной* наилучшей последовательности состояний, максимизирующей $P(Q|O,\lambda)$. Данная задача эквивалентна поиску максимума функции $P(Q,O|\lambda)$. Метод определения такой наилучшей последовательности состояний и есть алгоритм Витерби (16).

Пусть $\delta_t(i), \quad i=\overline{1,N}, \quad t=\overline{1,T}$ – это максимальная вероятность нахождения символа o_t в s_i состоянии. Функция $\psi_t(i), \quad i=\overline{1,N}, \quad t=\overline{1,T}$ указывает на предшествующие состояние s_{i-1}, которое максимизирует нахождение символа o_t в текущем состоянии s_i.

Согласно [26] алгоритм Витерби состоит из 4 этапов.

1. Инициализация:

$$\delta_1(i)=\pi_i b_i(O_1),\ i=\overline{1,N},$$

$$\psi_1(i)=0.$$

2. Рекурсивное вычисление:

$$\delta_t(j)=\max_{i=1,N}[\delta_{t-1}(i)a_{ij}]b_j(O_t),\ t=\overline{2,T}, \quad j=\overline{1,N}, \tag{1.16}$$

$$\psi_t(j)=\arg\max_{i=1,N}[\delta_{t-1}(i)a_{ij}],\ t=\overline{2,T}, \quad j=\overline{1,N}.$$

3. Завершение:

$$P^*=\max_{i=1,N}[\delta_T(i)],$$

$$q_T^*=\arg\max_{i=1,N}[\delta_T(i)].$$

4. Выполнение шагов для получения оптимальной последовательности состояний:

$$q_t^*=\psi_{t+1}(q_{t+1}^*),\ t=\overline{T-1,1}.$$

В результате мы получим оптимальную последовательность состояний $Q^*=q_1^*,q_2^*,...,q_T^*$ и P^* – вероятность того, что полученная последовательность Q

соответствует данной модели $\lambda = (A, B, \pi)$.

На практике при возрастании t значения $\psi_t(j)$ могут вызвать переполнение регистров или быть машинным нулем, и из-за этого будут возникать переполнения или потеря значимости при вычислениях. Для решения этой проблемы надо брать натуральный логарифм от всех вероятностей и плотностей.

Данный алгоритм является эффективным поиском наилучшей последовательности состояний: $P(O, Q^* \mid \lambda) = \max_{p \in Q^T} P(O, p \mid \lambda)$. Доказательство данного факта приведено в [15].

Более строго определение этого алгоритма для обучения (см. пункт 1.2.5) а так же некоторые изменения приведены в [17]. Использование алгоритма Витерби и его модификаций для задачи распознавания рассмотрено [16].

На рисунке 1.5 приведена схема, поясняющая работу данного алгоритма (при $\pi = (1,0,...,0)^T$). Большими кружками отмечены выбранные состояния для наблюдений o_t.

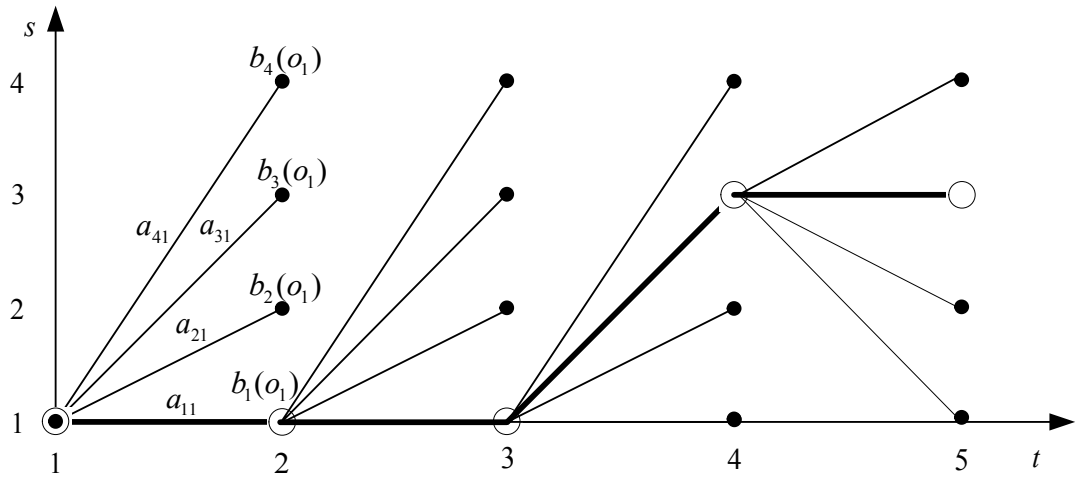

Рисунок 1.5. Схема работы алгоритма Витерби

Для каждой модели, в результате работы алгоритма Витерби, генерируется оптимальная последовательность состояний и вычисляется вероятность того, что

полученная последовательность соответствует данной модели $\lambda^{(r)}$, $r = \overline{1, R}$. Далее выбирается максимальная среди данных вероятностей величина:

$$P\left(O^{(test)} \mid \lambda_p\right) = \max_{r=\overline{1,R}} P\left(O^{(test)} \mid \lambda_r\right). \tag{1.17}$$

Таким образом, соответствующая данной вероятности модель укажет на персону, к которой относится тестовая фотография.

1.4. Выводы

В данной главе были рассмотрены теоретические аспекты этапов распознавания и обучения СММ. Был дан некоторый обзор источников литературы по данным вопросам.

Так же были получены следующий результат: кроме определения оптимальной последовательности скрытых состояний алгоритм Витерби увеличивает значение функции правдоподобия при обучении СММ. Подключение на последнем этапе обучения алгоритма Баум-Велша приводит к окончательному выводу значения функции правдоподобия на максимальное значение (см. рисунок 1.4).

В результате изучения алгоритмов кластеризации наблюдений на состояния мы пришли к выводу, что существенного влияния выбор одного из алгоритмов на результаты распознавания не оказывает.