

Скрытые марковские модели

§ 1.1. Марковские цепи и процессы

Марковская цепь задаётся начальным распределением вероятностей $p^0(x)$ и вероятностями перехода $\Gamma(x'; x)$. $\Gamma(x'; x)$ — это распределение следующего элемента цепи в зависимости от предыдущего; распределение на $(t + 1)$ -м шаге равно

$$p^{t+1}(x') = \int \Gamma(x'; x)p^t(x)dx.$$

В дискретном случае $\Gamma(x'; x)$ — это матрица вероятностей переходов между состояниями $p(x' = i|x = j)$.

Мы будем рассматривать только дискретные задачи. С помощью марковской модели можно моделировать марковский процесс, при этом подразумевается, что мы можем наблюдать какие-то функции от марковского процесса.

Скрытые марковские модели предполагают, что мы не можем обычно получить сами состояния, т.е. мы не знаем, сколько этих состояний и какие между ними связи, — это всё неизвестные параметры модели.

Нам известны лишь наблюдаемые величины $y(t)$, которые зависят от скрытых переменных $x(t)$ (см. рис. 1.1). Наша задача заключается в том, чтобы определить скрытые параметры процесса. Иными словами, по имеющимся данным $y(t)$ необходимо понять, каковы наиболее вероятные $x(t)$ и наиболее вероятная модель этого марковского процесса.

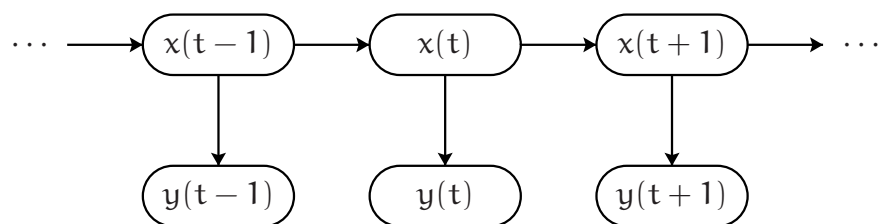


Рис. 1.1. Дискретный марковский процесс.

Главное свойство — следующее состояние зависит только от предыдущего и не зависит от истории. Формально это значит, что вероятность $x(t)$ при условии всех

остальных установленных значений равна вероятности $x(t)$ при условии только предыдущего значения:

$$p(x(t) = x_j | x(t-1) = x_{j_{t-1}}, \dots, x(1) = x_{j_1}) = p(x(t) = x_j | x(t-1) = x_{j_{t-1}}).$$

Боле того, вероятности $a_{ij} = p(x(t) = x_j | x(t-1) = x_i)$ ещё и от времени t не зависят. Эти вероятности составляют матрицу перехода $A = (a_{ij})$.

Для вероятностей перехода a_{ij} выполняются следующие естественные свойства:

- $a_{ij} \geq 0$;
- $\sum_j a_{ij} = 1$ (если мы начнём в состоянии i , то куда-нибудь мы точно попадём).

§ 1.2. Постановка задачи

Прямая задача заключается в определении того, с какой вероятностью выпадет та или иная последовательность событий $Q = q_{i_1} \dots q_{i_k}$. Если Q — это последовательность состояний марковской цепи, то отыскание её вероятности не составляет никакого труда:

$$p(Q|\text{модель}) = p(q_{i_1})p(q_{i_2}|q_{i_1}) \dots p(q_{i_k}|q_{i_{k-1}}).$$

Но реальные задачи гораздо сложнее. Сложность в том, что никто нам не скажет, что модель должна быть именно такой. Мы обычно наблюдаем не $x(t)$, т.е. реальные состояния модели, а $y(t)$, т.е. некоторую функцию от них (данные).

Давайте приведём пример.

П р и м е р 1.1. Подбрасывание монетки.

Предположим, что кто-то бросает монетку и сообщает нам результаты — последовательность орлов и решек. Но при этом подбрасывание монетки происходит где-то там, в другой комнате, поэтому мы только знаем, что есть определённая последовательность битов.

Если он бросает одну монетку, то модель будет выглядеть следующим образом (см. рис. 1.2а). Будет два состояния: 0 (выпал орёл) и 1 (выпала решка). С вероятностью $1-p$ выпадает орёл, с вероятностью p — решка. Это одна модель.

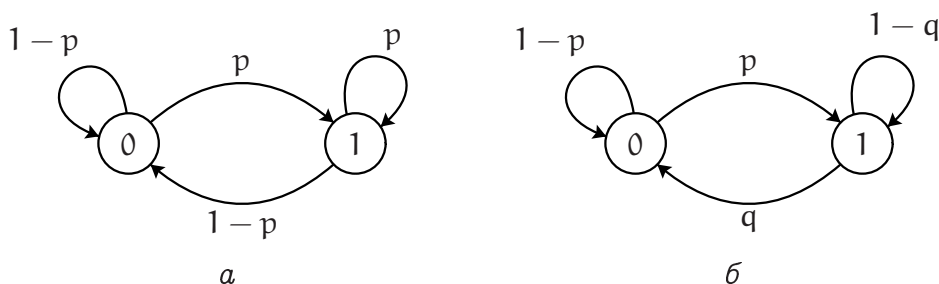


Рис. 1.2. Варианты моделей: *a* — модель с одной монеткой; *б* — модель с двумя монетками.

Но мы же можем подумать, что у него две монетки! Он кидает монетку. Когда у него выпадает решка, он переходит на другую монетку. Когда у него выпадает орёл на другой монетке, он возвращается к подбрасыванию первой монетки. В данном случае уже две вероятности: p и q . Модель уже совершенно другая (см. рис. 1.26). В ней по-прежнему два состояния, но параметров больше. А если три монетки?..

Наша задача в том числе и в том, чтобы понять, какая из этих моделей лучше соответствует известным данным.

Рассмотрим формулировки трёх основных задач.

1. Найти вероятность последовательности наблюдений в данной модели. Например, для модели с одной монеткой (рис. 1.2а) найти вероятность последовательности ОРРОО (О — орёл, Р — решка).
2. Найти «оптимальную» последовательность состояний при условии данной модели и данной последовательности наблюдений.
3. Найти наиболее правдоподобную модель (параметры модели). Например, найти параметры p и q для модели на рис. 1.2б.

§ 1.3. Обозначения в скрытых марковских моделях

Введём обозначения для состояний и наблюдаемых:

- $X = \{x_1, \dots, x_n\}$ — множество состояний;
- $V = \{v_1, \dots, v_m\}$ — алфавит, из которого мы выбираем наблюдаемые y (множество значений y);
- q_t — состояние во время t , y_t — наблюдаемая во время t .

Данные будем обозначать через $D = d_1 \dots d_T$ (последовательность наблюдаемых, d_i принимают значения из V).

Для вероятностей и распределений будем использовать следующие обозначения:

- $a_{ij} = p(q_{t+1} = x_j | q_t = x_i)$ — вероятность перехода из i в j ;
- $b_j(k) = p(v_k | x_j)$ — вероятность получить данные v_k в состоянии j ;
- начальное распределение $\pi = \{\pi_j\}$, $\pi_j = p(q_1 = x_j)$.

Модель $\lambda = (A, B, \pi)$ состоит из матрицы перехода A , матрицы наблюдаемых B и начального распределения π .

На рис. 1.3 показано, как работает скрытая марковская модель (Hidden Markov Model, НММ). Таким алгоритмом можно выбрать случайную последовательность наблюдаемых.

§ 1.4. Задачи формально

Теперь можно формализовать постановку задач.

HMME_{mulation}(λ):

1. Выберем начальное состояние x_1 по распределению π .
2. Для всех t от 1 до T :
 - а) Выберем наблюдаемую d_t по распределению $b_j(k) = p(v_k|x_j)$.
 - б) Выберем следующее состояние по распределению $a_{ij} = p(q_{t+1} = x_j|q_t = x_i)$.

Рис. 1.3. Алгоритм эмуляции скрытой марковской модели.

- Первая задача: по данной модели $\lambda = (A, B, \pi)$ и последовательности D найти $p(D|\lambda)$. Фактически, это нужно для того, чтобы оценить, насколько хорошо модель подходит к данным.
- Вторая задача: по данной модели λ и последовательности D найти «оптимальную» последовательность состояний $Q = q_1 \dots q_T$. Как и раньше, будет два решения: «побитовое» и общее.
- Третья задача: оптимизировать параметры модели $\lambda = (A, B, \pi)$ так, чтобы максимизировать $p(D|\lambda)$ при данном D (найти модель максимального правдоподобия). Эта задача — главная, в ней и заключается обучение скрытых марковских моделей.

Перейдём к рассмотрению решений этих задач.

§ 1.5. Первая задача

Формально, первая задача выглядит так. Требуется найти вероятность данных D при условии модели λ . Кроме явных переменных D у нас есть ещё скрытые переменные Q . Давайте сделаем их явными. Для этого нам придётся по ним просуммировать

$$p(D|\lambda) = \sum_Q p(D|Q, \lambda)p(Q|\lambda).$$

Вероятность $p(Q|\lambda)$ последовательности $Q = q_1 \dots q_T$ при условии λ — это вероятность того, что мы выберем первый элемент, а потом будем переходить от первого ко второму и т.д. Чтобы подсчитать вероятность D при условии последовательности Q , мы на каждом шаге вычисляем вероятности получения соответствующих данных. Каждая последовательность — это отдельное событие. Они не пересекаются, поэтому мы просто берём сумму по всем этим событиям:

$$p(D|\lambda) = \sum_Q p(D|Q, \lambda)p(Q|\lambda) = \sum_{q_1, \dots, q_T} b_{q_1}(d_1) \dots b_{q_T}(d_T) \pi_{q_1} a_{q_1 q_2} \dots a_{q_{T-1} q_T}.$$

DynamicHMMObservationSequenceProbability(λ, D):

1. Инициализировать $\alpha_1(i) = \pi_i b_i(d_1)$.

2. Шаг индукции:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^n \alpha_t(i) a_{ij} \right] b_j(d_{t+1}).$$

3. После того как дойдём до шага T , подсчитаем искомую вероятность:

$$p(D|\lambda) = \sum_{i=1}^n \alpha_T(i).$$

Рис. 1.4. Алгоритм нахождения вероятности последовательности наблюдений в данной модели.

Решение напоминает нам задачу маргинализации. Есть «большая» функция, которая представляется в виде «большого» произведения, и требуется её просуммировать по части переменных. Мы воспользуемся так называемой forward-backward procedure, по сути — вычислением на решётке, как в декодировании. Будем последовательно вычислять промежуточные величины вида

$$\alpha_t(i) = p(d_1 \dots d_t, q_t = x_i | \lambda),$$

т.е. искомые вероятности, но ещё с учётом текущего состояния. Величина $\alpha_t(i)$ показывает вероятность всех имеющихся данных вплоть до t и того, что мы пришли в состояние x_i на шаге t в данной модели.

Тогда искомая вероятность $p(D|\lambda)$ представляется в виде

$$p(D|\lambda) = \sum_{i=1}^n \alpha_T(i).$$

Вспомогательные величины $\alpha_t(i)$ можно вычислять с помощью динамического программирования (см. рис. § 1.5). Сначала инициализируем $\alpha_1(i)$ произведением начального распределения π_i и вероятности получить d_1 . Затем выполняем шаг индукции. Для вычисления $\alpha_{t+1}(j)$ необходимо подсчитать сумму по всем предыдущим состояниям вероятностей достичь этого состояния и перейти в состояние j , после чего умножить эту сумму на вероятность получить в j -м состоянии новое данное d_{t+1} . После того как дойдём до шага T , просуммируем и найдём величину $p(D|\lambda)$.

Фактически, это только прямой проход, обратный нам здесь не понадобился. Обратный проход вычислял бы условные вероятности $\beta_t(i) = p(d_{t+1} \dots d_T | q_t = x_i, \lambda)$ — вероятности получения цепочки $d_{t+1} \dots d_T$, начиная из состояния i на шаге t . Их

можно вычислить, проинициализировав $\beta_T(i) = 1$, а затем по индукции:

$$\beta_t(i) = \sum_{j=1}^n a_{ij} b_j(d_{t+1}) \beta_{t+1}(j).$$

Это нам пригодится чуть позже, при решении второй и третьей задачи.

§ 1.6. Вторая задача

Как мы уже упоминали, возможны два варианта.

1. Решать «побитово», отвечая на вопрос, каково наиболее вероятное состояние во время j .
2. Решать задачу нахождения наиболее вероятной последовательности состояний.

Эти варианты решения задачи принципиально различаются. Если взять наиболее вероятные состояния во время j и составить из них последовательность, то можем получить некорректную последовательность (содержит переход, вероятность которого равна 0).

Приведём побитовое решение задачи. Для этого нам потребуются вспомогательные переменные

$$\gamma_t(i) = p(q_t = x_i | D, \lambda).$$

$\gamma_t(i)$ — это вероятность того, что мы придём на t -м шаге в i -е состояние при условии имеющихся данных и заданной модели. Наша задача — найти

$$q_t = \operatorname{argmax}_{1 \leq i \leq n} \gamma_t(i), \quad 1 \leq t \leq T.$$

Для нахождения значения q_t , которое максимизирует $\gamma_t(i)$, применяется вариант min-sum алгоритма.

Выражаем $\gamma_t(i)$ через α и β :

$$\gamma_t(i) = \frac{\alpha_t(i) \beta_t(i)}{p(D|\lambda)} = \frac{\alpha_t(i) \beta_t(i)}{\sum_{i=1}^n \alpha_t(i) \beta_t(i)}.$$

На знаменатель можно не обращать внимания, поскольку нам нужен argmax .

Чтобы ответить на вопрос о наиболее вероятной последовательности, мы будем использовать уже знакомый алгоритм Витерби, то есть, по сути, то же самое динамическое программирование.

Теперь нам понадобятся новые вспомогательные переменные

$$\delta_t(i) = \max_{q_1, \dots, q_{t-1}} p(q_1 q_2 \dots q_t = x_i, d_1 d_2 \dots d_t | \lambda).$$

$\delta_t(i)$ — максимальная вероятность достичь состояния x_i на шаге t среди всех путей с заданными наблюдаемыми.

DynamicHMMOptimalStateSequence(λ, D):

1. Инициализировать $\delta_1(i) = \pi_i b_i(d_1)$, $\psi_1(i) = []$.

2. Шаг индукции:

$$\delta_t(j) = \max_{1 \leq i \leq n} [\delta_{t-1}(i) a_{ij}] b_j(d_t),$$

$$\psi_t(j) = \operatorname{argmax}_{1 \leq i \leq n} [\delta_{t-1}(i) a_{ij}].$$

3. После шага T вычислить:

$$p^* = \max_{1 \leq i \leq n} \delta_T(i), \quad q_T^* = \operatorname{argmax}_{1 \leq i \leq n} \delta_T(i).$$

4. Вычислить последовательность: $q_t^* = \psi_{t+1}(q_{t+1}^*)$.

Рис. 1.5. Алгоритм нахождения наиболее вероятной последовательности состояний.

По индукции получаем

$$\delta_{t+1}(j) = \left[\max_i \delta_t(i) a_{ij} \right] b_j(d_{t+1}).$$

Кроме того, потребуется запоминать аргументы, а не только значения; для этого будет массив $\psi_t(j)$.

На рис. § 1.6 приведён алгоритм решения задачи относительно последовательности. Сначала проинициализируем $\delta_1(i)$ начальными вероятностями, а $\psi_1(i)$ — пустым массивом. Затем по индукции вычисляем значения $\delta_t(j)$ и аргументы $\psi_t(j)$, на которых достигаются максимумы. Когда мы дойдём до шага T , подсчитаем p^* (оно нам не понадобится) и q_T^* . После этого, возвращаясь обратно по массиву ψ , будем считать аргументы.

§ 1.7. Третья задача

Задача состоит в нахождении параметров модели a_{ij} , $b_j(k)$, π_i при условии имеющихся данных.

Отметим, что аналитически найти глобальный максимум $p(D|\lambda)$ у нас никак не получится. Зато мы рассмотрим итеративную процедуру (по сути — градиентный подъём), которая приведёт к локальному максимуму этой функции. У функции может быть несколько локальных максимумов, поэтому на практике этот алгоритм следует запускать несколько раз, чтобы он попадал в разные элементы пространства и мог бы привести к разным максимумам. Алгоритм, который мы будем рассматривать, называется алгоритмом Баума–Велха (Baum–Welch algorithm). Он является на самом деле частным случаем алгоритма EM. При наличии некоторой модели с некоторыми параметрами согласно алгоритму EM (Estimation–Maximization algorithm)

мы сначала считаем ожидание данных при условии этой модели, потом поправляем модель при условии имеющихся данных. В ходе такого итеративного процесса мы приходим к наилучшей модели.

Теперь нашими вспомогательными переменными будут вероятности того, что мы во время t в состоянии x_i , а во время $t + 1$ — в состоянии x_j :

$$\xi_t(i, j) = p(q_t = x_i, q_{t+1} = x_j | D, \lambda).$$

Выражение для $\xi_t(i, j)$ можно переписать через уже знакомые переменные:

$$\xi_t(i, j) = \frac{\alpha_t(i) a_{ij} b_j(d_{t+1}) \beta_{t+1}(j)}{p(D|\lambda)} = \frac{\alpha_t(i) a_{ij} b_j(d_{t+1}) \beta_{t+1}(j)}{\sum_i \sum_j \alpha_t(i) a_{ij} b_j(d_{t+1}) \beta_{t+1}(j)}.$$

Отметим также, что $\gamma_t(i) = \sum_j \xi_t(i, j)$.

$\sum_t \gamma_t(i)$ — это ожидаемое количество переходов из состояния x_i , а $\sum_t \xi_t(i, j)$ — из x_i в x_j . Теперь на шаге M мы будем переоценивать вероятности:

$$\begin{aligned} \bar{\pi}_i &= \text{ожидаемая частота появления } x_i \text{ на шаге } 1 = \gamma_1(i), \\ \bar{a}_{ij} &= \frac{\text{количество переходов из } x_i \text{ в } x_j}{\text{количество переходов из } x_i} = \frac{\sum_t \xi_t(i, j)}{\sum_t \gamma_t(i)}, \\ \bar{b}_i(k) &= \frac{\text{количество появлений в } x_i \text{ и наблюдений } v_k}{\text{количество появлений в } x_i} = \frac{\sum_{t: d_t = v_k} \gamma_t(i)}{\sum_t \gamma_t(i)}. \end{aligned}$$

EM-алгоритм приведёт к цели: начинаем с какой-то модели $\lambda = (A, B, \pi)$, вычисляем вспомогательные переменные, подсчитываем $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$, снова пересчитываем параметры и т.д.

§ 1.8. Обоснование алгоритма Баума–Велха

Расстояние Кульбака–Лейблера (Kullback–Leibler distance, divergence) — это информационно-теоретическая мера того, насколько далеки распределения друг от друга:

$$D_{\text{KL}}(p_1, p_2) = \sum_x p_1(x) \log \frac{p_1(x)}{p_2(x)}.$$

Известно, что это расстояние всегда неотрицательно, равно нулю тогда и только тогда, когда p_1 и p_2 совпадают в каждой точке: $p_1 \equiv p_2$.

Определим распределения p_1 и p_2 на множествах последовательных состояний для моделей λ и λ' следующим образом:

$$p_1(Q) = \frac{p(Q, D|\lambda)}{p(D|\lambda)}, \quad p_2(Q) = \frac{p(Q, D|\lambda')}{p(D|\lambda')}.$$

Тогда расстояние Кульбака–Лейблера для распределений p_1 и p_2 :

$$\begin{aligned} 0 \leq D_{\text{LK}}(\lambda, \lambda') &= \sum_{\mathcal{Q}} p_1(Q) \log \frac{p_1(Q)}{p_2(Q)} = \sum_{\mathcal{Q}} \frac{p(Q, D|\lambda)}{p(D|\lambda)} \log \frac{p(Q, D|\lambda)p(D|\lambda')}{p(Q, D|\lambda')p(D|\lambda)} = \\ &= \log \frac{p(D|\lambda')}{p(D|\lambda)} + \sum_{\mathcal{Q}} \frac{p(Q, D|\lambda)}{p(D|\lambda)} \log \frac{p(Q, D|\lambda)}{p(Q, D|\lambda')}. \end{aligned}$$

Введём вспомогательную функцию

$$\mathcal{Q}(\lambda, \lambda') = \sum_{\mathcal{Q}} p(Q|D, \lambda) \log p(Q|D, \lambda').$$

Тогда из неравенства следует, что

$$\frac{\mathcal{Q}(\lambda, \lambda') - \mathcal{Q}(\lambda, \lambda)}{p(D|\lambda)} \leq \log \frac{p(D|\lambda')}{p(D|\lambda)}.$$

Если $\mathcal{Q}(\lambda, \lambda') > \mathcal{Q}(\lambda, \lambda)$, то $0 \leq \log \frac{p(D|\lambda')}{p(D|\lambda)}$, следовательно $p(D|\lambda') > p(D|\lambda)$. Значит, если мы максимизируем $\mathcal{Q}(\lambda, \lambda')$ по λ' , мы тем самым будем двигаться в нужную сторону.

Итак, требуется максимизировать $\mathcal{Q}(\lambda, \lambda')$ по λ' . Перепишем:

$$\begin{aligned} \mathcal{Q}(\lambda, \lambda') &= \sum_{\mathcal{Q}} p(Q|D, \lambda) \log p(Q|D, \lambda') = \sum_{\mathcal{Q}} p(Q|D, \lambda) \log \pi_{q_1} \prod_t a_{q_{t-1}q_t} b_{q_t}(d_t) = \\ &= \sum_{\mathcal{Q}} p(Q|D, \lambda) \log \pi_{q_1} + \sum_{\mathcal{Q}} p(Q|D, \lambda) \sum_t \log a_{q_{t-1}q_t} + \sum_{\mathcal{Q}} p(Q|D, \lambda) \sum_t \log b_{q_t}(d_t). \end{aligned}$$

Последнее выражение легко продифференцировать по a_{ij} , $b_i(k)$ и π_i , добавить соответствующие множители Лагранжа и решить. Покажем, что получится именно пересчёт по алгоритму Баума–Велха.

Поскольку параметры π_i , a_{ij} и $b_i(k)$, по которым выполняется оптимизация, независимо входят в каждое из трёх слагаемых выражения для $\mathcal{Q}(\lambda, \lambda')$, можно оптимизировать каждое из этих слагаемых по-отдельности.

Рассмотрим первое слагаемое:

$$\sum_{\mathcal{Q}} p(Q|D, \lambda) \log \pi_{q_1} = \sum_{i=1}^n p(q_1 = x_i | D, \lambda) \log \pi_i.$$

Добавляя множитель Лагранжа μ и используя ограничение $\sum_i \pi_i = 1$, запишем выражение для производной и приравняем его к нулю:

$$\frac{\partial}{\partial \pi_i} \left(\sum_{i=1}^n p(q_1 = x_i | D, \lambda) \log \pi_i + \mu \left(\sum_{i=1}^n \pi_i - 1 \right) \right) = 0.$$

Продифференцировав это выражение по π_i , просуммировав по i с использованием равенства $\sum_i \pi_i = 1$ для отыскания μ и решив относительно π_i , получаем

$$\pi_i = \frac{p(q_1 = x_i | D, \lambda)}{p(D | \lambda)}.$$

Перепишем второе слагаемое суммы $Q(\lambda, \lambda')$:

$$\sum_Q p(Q | D, \lambda) \sum_t \log a_{q_{t-1} q_t} = \sum_{i=1}^n \sum_{j=1}^n \sum_t p(q_t = x_i, q_{t+1} = x_j | D, \lambda) \log a_{ij}.$$

Вводя множитель Лагранжа и применяя ограничение $\sum_{j=1}^n a_{ij} = 1$, получаем

$$a_{ij} = \frac{\sum_t p(q_t = x_i, q_{t+1} = x_j | D, \lambda)}{\sum_t p(q_t = x_i | D, \lambda)} = \frac{\sum_t \xi_t(i, j)}{\sum_t \gamma_t(i)}.$$

Наконец, третье слагаемое суммы $Q(\lambda, \lambda')$ запишем в виде

$$\sum_Q p(Q | D, \lambda) \sum_t \log b_{q_t}(d_t) = \sum_{i=1}^n \sum_t p(q_t = x_i | D, \lambda) \log b_i(d_t).$$

Снова применяем метод множителей Лагранжа с ограничением $\sum_j b_i(j) = 1$. Заметим, что только наблюдаемые d_t , равные v_k , вносят вклад в результирующее значение для вероятности:

$$b_i(k) = \frac{\sum_{t: d_t = v_k} p(q_t = x_i | D, \lambda)}{\sum_t p(q_t = x_i | D, \lambda)} = \frac{\sum_{t: d_t = v_k} \gamma_t(i)}{\sum_t \gamma_t(i)}.$$

§ 1.9. Упражнения и задачи

1. Реализовать систему обучения скрытых марковских моделей, которая умела бы решать все три вышеописанные задачи.