Лекция 11
Скрытые марковские цепи для сегментации временных рядов

Артемов А. В.
мФТиАД ФКН ВШЭ

26 апреля 2018

1 Назад в прошлое: динамическое программирование

Для описания некоторых алгоритмов, которые появятся дальше, будет нужен метод динамического программирования. Рассмотрим его на примере *задачи объезда стран*:

Задача 1. *Путешественник хочет устроить турне: начиная с города x_s и заканчивая городм x_f, он хочет посетить k стран в определённой последовательности. Далее, он ночует в каждой стране, которую посещает. Однако у него мало денег, поэтому он хочет устроить турне с минимальными затратами. Какие города он должен посетить?*

Понятно, что решать эту задачу перебором слишком долго: если в каждой стране n городов, то придётся рассмотреть n^k вариантов поездок. Попробуем немного улучшить эту оценку.

Пусть города задаются следующим образом: x_{ij} означает «j-й город в i-й стране» (для этого занумеруем все страны и все города в них). Далее, пусть $f(x_{ij}, x_{i+1,k})$ — стоимость проезда из города x_{ij} в город $x_{i+1,k}$, а $h(x_{ij})$ — стоимость ночлега в городе x_{ij}. Пользуясь этими обозначениями, введём *функцию Беллмана* $V(x)$ по следующему правилу:

$$V(x_s) = 0, \quad V(x_{i+1,k}) = \min_j \left[V(x_{ij}) + f(x_{ij}, x_{i+1,k}) + h(x_{i+1,k}) \right].$$

Оказывается, что смысл функции Беллмана $V(x)$ — это наименьшая сумма, которую нужно затратить для того, чтобы добраться из города x_s в город x. Это утверждение несложно доказать, пользуясь индукцией.

Пользуясь этой функцией, несложно получить решение. Введём функцию $S(x)$, действующую по правилу

$$S(x_{i+1,k}) = \arg\min_j \left[V(x_{ij}) + f(x_{ij}, x_{i+1,k}) + h(x_{i+1,k}) \right].$$

Тогда оптимальный путь $(x_1^*, \ldots, x_k^*)$ может быть построен рекурсивным вызовом функции S: $x_k^* = S(x_f)$, $x_m^* = S(x_{m+1}^*)$.

Решение задачи объезда стран показывает пример применения метода динамического программирования. Но, понятное дело, этот метод далеко не универсален. Когда его применение оправдано?

- Когда для решения задачи нужно решить перекрывающиеся подзадачи и их решения можно запомнить.

- Когда задача обладает оптимальной подструктурой, или же кусочной оптимальностью. Это означает следующее: например, если известно, что оптимальный путь из A в B содержит C и D и известен оптимальный путь из C в D, то этот путь станет частью оптимального пути из A в B.

2 Мотивация и основные понятия

Допустим, что мы хотим определить среднюю годовую температуру в каком-то месте на Земле за определённый период времени. Далее, усложним задачу тем, что возьмём очень давние времена — когда ещё не существовало термометров. К сожалению, отправиться назад в прошлое и замерить температуру не представляется возможным, поэтому придётся ориентироваться на косвенные признаки определённых температур.

Для простоты скажем, что температуру мы измеряем очень просто — различаются лишь состояния «горячо» и «холодно». Предположим, что современные измерения показывают, что тёплый год следует за тёплым годом с вероятностью 0,7, а холодный за холодным — с вероятностью 0,6. Далее предположим (на не самом понятном основании, но не суть), что это предположение имеет место и для давних времён. Тогда переходная матрица будет устроена следующим образом (Г — горячо, Х – холодно):

$$\begin{array}{c} \\ \text{Г} \\ \text{Х} \end{array} \begin{array}{c} \begin{array}{cc} \text{Г} & \text{Х} \end{array} \\ \begin{pmatrix} 0{,}7 & 0{,}3 \\ 0{,}4 & 0{,}6 \end{pmatrix} \end{array}$$

Преположим, что исследование показало зависимость диаметра древесных колец от средней годовой температуры. Опять же, для простоты будем различать только три размера — маленькие (М), средние (С) и большие (Б). Напоследок предположим, что исследование показало следующую вероятностную зависимость между ними:

$$\begin{array}{c} \\ \text{Г} \\ \text{Х} \end{array} \begin{array}{c} \begin{array}{ccc} \text{М} & \text{С} & \text{Б} \end{array} \\ \begin{pmatrix} 0{,}1 & 0{,}4 & 0{,}5 \\ 0{,}7 & 0{,}2 & 0{,}1 \end{pmatrix} \end{array}$$

Для данной системы *состоянием* служит среднегодовая температура — либо «горячо», либо «холодно». При этом переход из текущего состояния в следующее является марковским (первого порядка[1]), так как мы предположили, что он зависит только от текущего состояния и полностью задаётся переходной матрицей. Однако настоящее состояние «спрятано», так как мы не можем его определить напрямую.

Хоть мы и не можем наблюдать состояние (температуру) в прошлом, но мы можем замерить размер древесных колец. Вероятностная зависимость между диаметром колец и температурой даёт нам информацию о состоянии. Так как состояния «скрыты», то такие системы называют *скрытыми марковскими моделями*.

Теперь можно дать формальное определение.

Определение 1. Скрытая марковская модель (первого порядка) — это вероятностная модель последовательности, которая состоит из набора *наблюдаемых* переменных $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, где $\mathbf{x}_k \in \mathbb{R}^d$, и набора *латентных* (или *скрытых*) переменных

$$\mathbf{T} = \{\mathbf{t}_1, \ldots, \mathbf{t}_n\}, \quad \mathbf{t}_k \in \{0,1\}^K, \quad \sum_{i=1}^{K} t_{ki} = 1.$$

[1]Марковская цепь называется цепью k-го порядка, если состояние зависит от k предыдущих состояний. По умолчанию будем считать, что все рассматриаемые нами модели имеют первый порядок.

В данной модели латентные переменные являются *бинарными* и кодируют K состояний, поэтому их иногда называют переменными состояния. Значения наблюдаемого вектора $\mathbf{x}_k$, взятого в момент времени k, зависит только от скрытого состояния $\mathbf{t}_k$, которое, в свою очередь, зависит только от скрытого состояния в предыдущий момент времени $\mathbf{t}_{k-1}$.

Скрытые марковские модели имеют множество применений: распознание речи, образов на видео, поведения, анализирование фондовых рынков, естественных языков, ДНК и так далее.

Теперь опишем то, как можно задать некоторые параметры вероятносной модели. Начнём с распределения $\mathbf{t}_n$. Пусть в скрытой марковской модели K состояний. Закодируем их состояния в момент времени n бинарным вектором $\mathbf{t}_n = (t_{n1}, \ldots, t_{nK})$ по правилу:

$$t_{ij} = \begin{cases} 1, & \text{система находится в состоянии } j \text{ в момент времени } i, \\ 0, & \text{иначе.} \end{cases}$$

Так как в векторе $\mathbf{t}_n$ может быть только один ненулевой элемент (предполагается, что система не может находиться в двух разных состояниях одновременно), то распределение $\mathbf{t}_n$ относительно $\mathbf{t}_{n-1}$ $p(\mathbf{t}_n \mid \mathbf{t}_{n-1})$ можно задать матрицей $\mathbf{A}$, где $\mathbf{A}_{ij} = p(t_{nj} = 1 \mid t_{n-1,i} = 1)$. Стоит заметить, что $\sum_{j=1}^{K} \mathbf{A}_{ij} = 1$. Следовательно, распределение можно записать следующим образом:

$$p(\mathbf{t}_n \mid \mathbf{t}_{n-1}) = \prod_{i=1}^{K} \prod_{j=1}^{K} \mathbf{A}_{ij}^{t_{n-1,i} t_{nj}}.$$

Далее, нужно задать начальное распределение $p(\mathbf{t}_1)$. Пусть $\pi_i = p(t_{1i} = 1)$. Тогда

$$p(\mathbf{t}_1) = \prod_{i=1}^{K} \pi_i^{t_{1i}}.$$

Хоть матрица $\mathbf{A}$ может быть почти любой (нужна только неотрицательность элементов и равенство единице сумме элементов в строке), но с точки зрения скрытых марковских моделей более интересны матрицы $\mathbf{A}$ с преобладающими элементами на диагонали (то есть более вероятно то, что система не изменит своего состояния). В этом случае можно сказать, что процесс находится в одном и том же состоянии в течение какого-то отрезка времени. Отсюда получаем простую физическую интерпретацию скрытой марковской модели: это процесс, который иногда меняет свои характеристики.

Теперь вспомним, что наблюдаемая переменная $\mathbf{x}_n$ зависит только от переменной состояния $\mathbf{t}_n$. Следовательно, разумно рассматривать условное распределение $p(\mathbf{x}_n \mid \mathbf{t}_n)$. Обычно предполагается, что оно известно с точностью до параметров $\varphi_k, k \in \{1, \ldots, K\}$: то есть, если $t_{ni} = 1$, то $p(\mathbf{x}_n \mid \mathbf{t}_n) = p(\mathbf{x}_n \mid \varphi_i)$. Следовательно,

$$p(\mathbf{x}_n \mid \mathbf{t}_n) = \prod_{k=1}^{K} p(\mathbf{x}_n \mid \varphi_k)^{t_{nk}}.$$

Введённых выше параметров достаточно для полного описания скрытой марковской модели. Их собирают в *набор параметров*

$$\Theta = (\boldsymbol{\pi}, \mathbf{A}, \boldsymbol{\varphi}), \text{ где } \boldsymbol{\pi} = (\pi_1, \ldots, \pi_K), \quad \boldsymbol{\varphi} = (\varphi_1, \ldots, \varphi_K).$$

Теперь можно сформулировать основные задачи теории скрытых марковских процессов:

- *Обучение с учителем.* Пусть есть некоторая последовательность $\mathbf{X}$, для которой известны латентные переменные $\mathbf{T}$. По обучающей выборке нужно оценить набор параметров Θ.
- *Сегментация.* Пусть известна последовательность наблюдаемых переменных $\mathbf{X}$ и набор параметров Θ. По ним нужно построить максимально правдоподобный набор латентных переменных $\mathbf{T}$, то есть найти $\arg\max_{\mathbf{T}} p(\mathbf{T} \mid \mathbf{X}, \Theta)$.
- *Обучение без учителя.* Пусть известна последовательность наблюдаемых переменных $\mathbf{X}$ и число состояний K. Нужно оценить набор параметров Θ. Подзадача — *нахождение маргинального распределения*: найти $p(t_n \mid \mathbf{X}, \Theta)$.
- *Прогнозирование.* Пусть известна некоторая последовательность $\mathbf{X}$ длины N. Нужно оценить наблюдаемый вектор в момент времени $N+1$, то есть найти $p(\mathbf{x}_{N+1} \mid \mathbf{X})$.

3 Пример 1: обучение с учителем

Пусть дана обучающая выборка $(\mathbf{X}, \mathbf{T})$, представляющая собой одну или несколько последовательностей, в которых известны значения скрытых компонент. По ней нужно оценить набор параметров Θ. Как это сделать?

Для начала посмотрим на вероятность того, что обучающая выборка «соответствует» набору, то есть на $p(\mathbf{X}, \mathbf{T} \mid \Theta)$. По построению скрытой марковской модели это распределение задаётся следующим образом:

$$p(\mathbf{X}, \mathbf{T} \mid \Theta) = p_{\boldsymbol{\pi}}(\mathbf{t}_1) \prod_{k=1}^{N} p_{\boldsymbol{\varphi}}(\mathbf{x}_k \mid \mathbf{t}_k) \prod_{k=2}^{N} p_{\mathbf{A}}(\mathbf{t}_k \mid \mathbf{t}_{k-1}).$$

Подставляем ранее выписанные формулы для $p_{\boldsymbol{\pi}}(\mathbf{t}_1)$, $p_{\boldsymbol{\varphi}}(\mathbf{x}_k \mid \mathbf{t}_k)$ и $p_{\mathbf{A}}(\mathbf{t}_k \mid \mathbf{t}_{k-1})$:

$$p(\mathbf{X}, \mathbf{T} \mid \Theta) = \prod_{i=1}^{K} \pi_i^{t_{1i}} \left(\prod_{n=1}^{N} \prod_{k=1}^{K} p(\mathbf{x}_n \mid \varphi_k)^{t_{nk}} \right) \left(\prod_{n=2}^{K} \prod_{i=1}^{K} \prod_{j=1}^{K} \mathbf{A}_{ij}^{t_{n-1,i} t_{nj}} \right)$$

Оценивать Θ будем с помощью метода максимального правдоподобия, то есть

$$\Theta_{ML} = \arg\max_{\Theta} p(\mathbf{X}, \mathbf{T} \mid \Theta) = \arg\max_{\Theta} \log p(\mathbf{X}, \mathbf{T} \mid \Theta)$$

Второе равенство корректно, так как $\log x$ — монотонно возрастающая функция. Теперь выпишем $\log p(\mathbf{X}, \mathbf{T} \mid \Theta)$:

$$\log p(\mathbf{X}, \mathbf{T} \mid \Theta) = \sum_{i=1}^{K} t_{1i} \log \pi_i + \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \log p(\mathbf{x}_n \mid \varphi_k) + \sum_{n=2}^{N} \sum_{i=1}^{K} \sum_{j=1}^{K} t_{n-1,i} t_{nj} \log \mathbf{A}_{ij}.$$

Для нахождения оценок воспользуемся методом Лагранжа. Вспомним, что на параметры наложены следующие ограничения:

$$\sum_{i=1}^{K} \pi_i = 1, \quad \forall i \in \{1, 2, \ldots, K\} \ \sum_{j=1}^{K} A_{ij} = 1.$$

Выпишем лагранжиан:

$$\mathcal{L}(\Theta, \lambda, \boldsymbol{\mu}) = \log p(\mathbf{X}, \mathbf{T} \mid \Theta) + \lambda \left(\sum_{i=1}^{K} \pi_i - 1 \right) + \sum_{i=1}^{K} \mu_i \left(\sum_{j=1}^{K} A_{ij} - 1 \right) \to \text{extr}.$$

Теперь начнём искать оценки.

- Начнём с оценки $\boldsymbol{\pi}$. Для неё

$$\frac{\partial \mathcal{L}(\Theta, \lambda, \boldsymbol{\mu})}{\partial \pi_i} = \frac{t_{1i}}{\pi_i} + \lambda = 0 \implies \pi_i = -\frac{t_{1i}}{\lambda}.$$

  Теперь вспомним ограничения на $\boldsymbol{\pi}$ и на $\mathbf{t}_1$:

$$\sum_{i=1}^{K} \pi_i = -\frac{1}{\lambda} \sum_{i=1}^{K} t_{1i} = -\frac{1}{\lambda} = 1 \implies \pi_i = t_{1i}.$$

- Теперь оценим матрицу $\mathbf{A}$:

$$\frac{\partial \mathcal{L}(\Theta, \lambda, \boldsymbol{\mu})}{\partial \mathbf{A}_{ij}} = \sum_{n=2}^{N} \frac{t_{n-1,i} t_{nj}}{A_{ij}} + \mu_i = 0 \implies A_{ij} = \sum_{n=2}^{N} \frac{t_{n-1,i} t_{nj}}{\mu_i}.$$

  Осталось избавиться от μ_i.

$$\sum_{j=1}^{K} A_{ij} = 1 \implies \sum_{j=1}^{K} \sum_{n=2}^{N} \frac{t_{n-1,i} t_{nj}}{\mu_i} = \sum_{n=2}^{N} \frac{t_{n-1,i}}{\mu_i} = 1 \implies \mu_i = \sum_{n=2}^{N} t_{n-1,i}$$

  Отсюда получаем оценку:

$$\mathbf{A}_{ij} = \frac{\sum_{n=2}^{N} t_{n-1,i} t_{nj}}{\sum_{n=2}^{N} t_{n-1,i}}.$$

- Осталось оценить $\boldsymbol{\varphi}$. Снова возьмём частную производную:

$$\frac{\partial \mathcal{L}(\Theta, \lambda, \boldsymbol{\mu})}{\partial \varphi_i} = \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \frac{\partial \log p(\mathbf{x}_n \mid \varphi_k)}{\partial \varphi_i} = \sum_{n\,:\,t_{ni}=1} \frac{\partial \log p(\mathbf{x}_n \mid \varphi_i)}{\partial \varphi_i} = 0.$$

  Получилась классическая задача на максимизацию правдоподобия по выборке iid объектов. Получаем, что

$$\varphi_i = \arg\max \sum_{n\,:\,t_{ni}=1} \log p(\mathbf{x}_n \mid \varphi_i).$$

  Для нахождения этого решения можно воспользоваться методами восстановления плотностей. Об одном из них, EM-алгоритме, будет рассказано позднее.

4 Пример 2: сегментация. Алгоритм Витерби

Пусть нам известна последовательность наблюдаемых переменных $\mathbf{X}$ и набор параметров скрытой марковской модели Θ. Как по ним построить максимально правдоподобный набор латентных переменных $\mathbf{T}$?

Как мы сказали ранее, нужный набор равен

$$\arg\max_{\mathbf{T}} p(\mathbf{T} \mid \mathbf{X}, \Theta).$$

Теперь надо заметить, что $p(\mathbf{X} \mid \Theta)$ не зависит от $\mathbf{T}$. Тогда можно провести следующую цепочку равенств:

$$\arg\max_{\mathbf{T}} p(\mathbf{T} \mid \mathbf{X}, \Theta) = \arg\max_{\mathbf{T}} \frac{p(\mathbf{X}, \mathbf{T} \mid \Theta)}{p(\mathbf{X} \mid \Theta)} = \arg\max_{\mathbf{T}} p(\mathbf{X}, \mathbf{T} \mid \Theta) = \arg\max_{\mathbf{T}} \log p(\mathbf{X}, \mathbf{T} \mid \Theta).$$

Оказывается, что это есть ни что иное, как задача динамического программирования! Действительно, вспомним задачу об объезде стран. Посмотрим на логарифм:

$$\log p(\mathbf{X}, \mathbf{T} \mid \Theta) = \sum_{i=1}^{K} t_{1i} \log \pi_i + \sum_{n=2}^{N} \sum_{i=1}^{K} \sum_{j=1}^{K} t_{n-1,i} t_{nj} \log \mathbf{A}_{ij} + \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \log p(\mathbf{x}_n \mid \varphi_k).$$

По аналогии с задачей, первое слагаемое определяет «пункт отбытия», слагаемые второй суммы — стоимость переезда из одной страны в другую, а слагаемые третьей — «стоимость ночлега» в выбранном городе. Составим функцию Беллмана:

$$V(t_{1j}) = \log \pi_j, \quad V(t_{nj}) = \max_i \left[V(t_{n-1,i}) + t_{n-1,i} t_{nj} \log \mathbf{A}_{ij} + t_{nj} \log p(\mathbf{x}_n \mid \varphi_j) \right]$$

Теперь, как и раньше, определяем функцию S по правилу

$$S(t_{1j}) = \varnothing, \quad S(t_{nj}) = \arg\max_i \left[V(t_{n-1,i}) + t_{n-1,i} t_{nj} \log \mathbf{A}_{ij} + t_{nj} \log p(\mathbf{x}_n \mid \varphi_j) \right]$$

Выполнив прямой обход по сигналу, мы определим значения $V(t_{ij})$ и $S(t_{ij})$. Далее, выполнив обратный проход, мы получим номера оптимальных состояний $(i^*(1), \ldots, i^*(N))$ по следующему правилу:

$$i^*(N) = \arg\max_i V(t_{Ni}), \quad i^*(k) = S(t_{k+1,i^*(k+1)})$$

Легко понять, что $\mathbf{t}_k$ определяются так: $t_{k,i^*(k)} = 1$, $t_{kl} = 0$ при $l \neq i^*(k)$.

Чем так хорош этот алгоритм? Он позволяет достаточно быстро производить сегментацию очень длинных сигналов. Более того, при некоторых модификациях он может делать это в реальном времени (с небольшой задержкой, конечно).

5 Пример 3: обучение без учителя. EM-алгоритм

Преположим, что есть некоторая графическая модель, в которой известна лишь часть значений переменных, при этом атомарные распределения известны с точностью до набора параметров Θ. Нужно оценить Θ по наблюдаемым величинам методом максимального правдоподобия, то есть найти

$$\Theta_{ML} = \arg\max_{\Theta} p(\mathbf{X} \mid \Theta).$$

Это называют *методом неполного правдоподобия*. По правилу суммирования неполное правдоподобие может быть получено суммированием по скрытым переменных полного правдоподобия, то есть

$$p(\mathbf{X} \mid \Theta) = \sum_{\mathbf{T}} p(\mathbf{X}, \mathbf{T} \mid \Theta).$$

Для многих моделей (в частности, для байесовских сетей) полное правдоподобие считается достаточно просто.

Далее, для удобства часто переходят к логарифму, что возможно из-за строгой монотонности $\log x$. В частности, раньше мы получили явные формулы для $\arg\max_\Theta p(\mathbf{X}, \mathbf{T} \mid \Theta) = \arg\max_\Theta \log p(\mathbf{X}, \mathbf{T} \mid \Theta)$. Вот здесь и возникает первый подводный камень. Из-за возникающего «логарифма суммы» оптимизация становится крайне сложной.

Теперь построим итеративный алгоритм для нахождения Θ_{ML}, называемый *EM-алгоритмом* (expectation-maximization algorithm). Смысл названия будет понятен немного

позднее. Для начала введём обозначение для логарифмической функции правдоподобия: $L(\Theta) = \log p(\mathbf{X} \mid \Theta)$.

Пусть после n-й итерации алгоритма мы получили значение Θ_n. Так как мы хотим максимизировать $L(\Theta)$, то мы хотим получить значение Θ такое, что разность

$$L(\Theta) - L(\Theta_n) = \log p(\mathbf{X} \mid \Theta) - \log p(\mathbf{X} \mid \Theta_n).$$

была максимальной. Заметьте, что мы пока что никак не трогали латентные или утерянные переменные. Если они есть, то EM-алгоритм предлагает естественный метод их включения. Периодически латентные переменные можно ввести просто как «трюк» для упрощения нахождения оценки максимального правдоподобия. В этом случае предполагается, что знание скрытых переменных успрощает анализ функции правдоподобия.

Пусть $\mathbf{T}$ — вектор скрытых переменных. Тогда по формуле полной вероятности

$$p(\mathbf{X} \mid \Theta) = \sum_{\mathbf{T}} p(\mathbf{X}, \mathbf{T} \mid \Theta) = \sum_{\mathbf{T}} p(\mathbf{X} \mid \mathbf{T}, \Theta) p(\mathbf{T} \mid \Theta).$$

Тогда разность лог-функций правдоподобия можно записать в следующем виде:

$$L(\Theta) - L(\Theta_n) = \log \left(\sum_{\mathbf{T}} p(\mathbf{X} \mid \mathbf{T}, \Theta) p(\mathbf{T} \mid \Theta) \right) - \log p(\mathbf{X} \mid \Theta_n)$$

Мы вернулись к логарифму суммы. Для того, чтобы избавиться от него, воспользуемся *неравенством Йенсена*, которое доказывалось в курсе математического анализа.

Теорема 1. *Пусть f — выпуклая (вниз) функция, определённая на интервале (a, b). Тогда для любых $x_1, \ldots, x_n \in (a, b)$ и $\lambda_1, \ldots, \lambda_n > 0$ таких, что $\lambda_1 + \ldots + \lambda_n = 1$, выполнено следующее неравенство:*

$$f \left(\sum_{k=1}^{n} \lambda_k x_k \right) \leqslant \sum_{k=1}^{n} \lambda_k f(x_k).$$

Если функция f выпукла вверх, то знак неравенства меняется на противоположный.

Применим это неравенство следующим образом. Пусть $\lambda_i = p(\mathbf{T} \mid \mathbf{X}, \Theta_n)$. Несложно понять, что такие константы подходят. Далее, логарифм — выпуклая (вверх) функция, поэтому использование неравенства легально. Тогда «подгоним» выражение внутри логарифма под эти константы:

$$\begin{aligned}
L(\Theta) - L(\Theta_n) &= \log \left(\sum_{\mathbf{T}} p(\mathbf{X} \mid \mathbf{T}, \Theta) p(\mathbf{T} \mid \Theta) \right) - \log p(\mathbf{X} \mid \Theta_n) = \\
&= \log \left(\sum_{\mathbf{T}} p(\mathbf{X} \mid \mathbf{T}, \Theta) p(\mathbf{T} \mid \Theta) \frac{p(\mathbf{T} \mid \mathbf{X}, \Theta_n)}{p(\mathbf{T} \mid \mathbf{X}, \Theta_n)} \right) - \log p(\mathbf{X} \mid \Theta_n) = \\
&= \log \left(\sum_{\mathbf{T}} p(\mathbf{T} \mid \mathbf{X}, \Theta_n) \frac{p(\mathbf{X} \mid \mathbf{T}, \Theta) p(\mathbf{T} \mid \Theta)}{p(\mathbf{T} \mid \mathbf{X}, \Theta_n)} \right) - \log p(\mathbf{X} \mid \Theta_n) \geqslant \\
&\geqslant \sum_{\mathbf{T}} p(\mathbf{T} \mid \mathbf{X}, \Theta_n) \log \left(\frac{p(\mathbf{X} \mid \mathbf{T}, \Theta) p(\mathbf{T} \mid \Theta)}{p(\mathbf{T} \mid \mathbf{X}, \Theta_n)} \right) - \log p(\mathbf{X} \mid \Theta_n) = \\
&= \sum_{\mathbf{T}} p(\mathbf{T} \mid \mathbf{X}, \Theta_n) \left(\log \left(\frac{p(\mathbf{X} \mid \mathbf{T}, \Theta) p(\mathbf{T} \mid \Theta)}{p(\mathbf{T} \mid \mathbf{X}, \Theta_n)} \right) - \log p(\mathbf{X} \mid \Theta_n) \right) = \\
&= \sum_{\mathbf{T}} p(\mathbf{T} \mid \mathbf{X}, \Theta_n) \log \left(\frac{p(\mathbf{X} \mid \mathbf{T}, \Theta) p(\mathbf{T} \mid \Theta)}{p(\mathbf{X} \mid \Theta_n) p(\mathbf{T} \mid \mathbf{X}, \Theta_n)} \right) \equiv \Delta(\Theta \mid \Theta_n).
\end{aligned}$$

В результате получаем, что $L(\Theta) \geqslant L(\Theta_n) + \Delta(\Theta \mid \Theta_n)$. Для удобства введём функцию $l(\Theta \mid \Theta_n) \equiv L(\Theta_n) + \Delta(\Theta \mid \Theta_n)$. Тогда наше неравенство имеет очень простой вид: $L(\Theta) \geqslant l(\Theta \mid \Theta_n)$.

Теперь у нас есть функция $l(\Theta \mid \Theta_n)$, ограниченная сверху лог-функцией правдоподобия $L(\Theta)$. Заметим, что $l(\Theta_n \mid \Theta_n) = L(\Theta_n)$:

$$l(\Theta_n) = L(\Theta_n) + \sum_{\mathbf{T}} p(\mathbf{T} \mid \mathbf{X}, \Theta_n) \log \left(\frac{p(\mathbf{X} \mid \mathbf{T}, \Theta_n) p(\mathbf{T} \mid \Theta_n)}{p(\mathbf{T} \mid \mathbf{X}, \Theta_n) p(\mathbf{X} \mid \Theta_n)} \right) =$$
$$= L(\Theta_n) + \sum_{\mathbf{T}} p(\mathbf{T} \mid \mathbf{X}, \Theta_n) \log \left(\frac{p(\mathbf{X}, \mathbf{T} \mid \Theta_n)}{p(\mathbf{X}, \mathbf{T} \mid \Theta_n)} \right) = L(\Theta_n).$$

Наша цель состоит в выборе значения Θ такого, что $L(\Theta)$ максимально. Мы показали, что функция $l(\Theta \mid \Theta_n)$ ограничена сверху $L(\Theta)$ и равенство достигается при текущей оценке, то есть при $\Theta = \Theta_n$. Следовательно, если Θ увеличивает $l(\Theta \mid \Theta_n)$, то она увеличивает и $L(\Theta)$. Для получения максимального прироста в значении $L(\Theta)$, EM-алгоритм ищет Θ такое, что $l(\Theta \mid \Theta_n)$ максимально. Это значение обозначают через Θ_{n+1}. Теперь посчитаем значение Θ_{n+1}:

$$\Theta_{n+1} \equiv \arg\max_{\Theta} l(\Theta \mid \Theta_n) =$$
$$= \arg\max_{\Theta} \left[L(\Theta_n) + \sum_{\mathbf{T}} p(\mathbf{T} \mid \mathbf{X}, \Theta_n) \log \left(\frac{p(\mathbf{X} \mid \mathbf{T}, \Theta) p(\mathbf{T} \mid \Theta)}{p(\mathbf{X} \mid \Theta_n) p(\mathbf{T} \mid \mathbf{X}, \Theta_n)} \right) \right]$$

Выбросим все члены, не зависящие от Θ — они не повлияют на значение Θ_{n+1}:

$$\Theta_n = \arg\max_{\Theta} \left[\sum_{\mathbf{T}} p(\mathbf{T} \mid \mathbf{X}, \Theta_n) \log \left(p(\mathbf{X} \mid \mathbf{T}, \Theta) p(\mathbf{T} \mid \Theta) \right) \right] =$$
$$= \arg\max_{\Theta} \left[\sum_{\mathbf{T}} p(\mathbf{T} \mid \mathbf{X}, \Theta_n) \log p(\mathbf{X}, \mathbf{T} \mid \Theta) \right] =$$
$$= \arg\max_{\Theta} \left[\mathsf{E}_{\mathbf{T} \mid \mathbf{X}, \Theta_n} [\log p(\mathbf{X}, \mathbf{T} \mid \Theta)] \right].$$

В формуле появляется $p(\mathbf{T} \mid \mathbf{X}, \Theta_n)$. Возникает вопрос: а как это посчитать? Для этого воспользуемся формулой Байеса:

$$p(\mathbf{T} \mid \mathbf{X}, \Theta_n) = \frac{p(\mathbf{X}, \mathbf{T} \mid \Theta_n)}{p(\mathbf{X} \mid \Theta_n)} = \frac{p(\mathbf{X}, \mathbf{T} \mid \Theta_n)}{\sum_{\mathbf{T}} p(\mathbf{X}, \mathbf{T} \mid \Theta_n)}.$$

Отсюда и получается название: мы *максимизируем* (M) условное математическое *ожидание* (E). Теперь можно записать алгоритм формально:

Алгоритм 1 EM-алгоритм

Вход: Набор наблюдаемых переменных $\mathbf{X}$
Выход: Набор параметров Θ_{ML}.

1: Инициализируем Θ_1 каким-либо образом.
2: $n \leftarrow 1$.
3: **while** алгоритм не сошёлся **do**
4: E-шаг: считаем $\mathsf{E}_{\mathbf{T} \mid \mathbf{X}, \Theta_n} [\log p(\mathbf{X}, \mathbf{T} \mid \Theta)]$
5: M-шаг: $\Theta_{n+1} \leftarrow \arg\max_{\Theta} \left[\mathsf{E}_{\mathbf{T} \mid \mathbf{X}, \Theta_n} [\log p(\mathbf{X}, \mathbf{T} \mid \Theta)] \right]$
6: $n \leftarrow n + 1$

По сути, этот алгоритм есть ни что иное, как покоординатный спуск: на каждой итерации последовательно уточняются возможные значения $\mathbf{T}$ (E-шаг), после чего пересчитывается значение Θ (M-шаг).

После всех логических изысканий можно задать вопрос: а зачем всё это? Почему бы не максимизировать $L(\Theta)$ сразу, а не ворочаться с $l(\Theta \mid \Theta_n)$? Ответ достаточно прост: $l(\Theta \mid \Theta_n)$ учитывает скрытые переменные T. В случае, когда мы хотим их определить, EM-алгоритмы предоставляют необходимые для этого инструменты. Более того, как мы говорили ранее, введение скрытых переменных может быть весьма удобным в том плане, что оптимизация $l(\Theta \mid \Theta_n)$ будет гораздо проще прямой оптимизации $L(\Theta)$. Например, во многих случаях на M-шаге можно получить явные формулы, так как происходит оптимизация выпуклой комбинации логарифмов полных правдоподобий.

Рассмотрим пример применения EM-алгоритма на задаче *разделения гауссовской смеси*. Она появляется при попытке приблизить плохо параметризуемые распределения смесью гауссиан. Пусть

$$\mathbf{X} \text{ — выборка размера } n \text{ из смеси } \sum_{k=1}^{l} w_k \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \in \mathbb{R}^d, \quad \sum_{k=1}^{l} w_k = 1.$$

Задача следующая: нужно восстановить плотность генеральной совокупности, то есть определить $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ и w_k. Попробуем применить EM-алгоритм. Для начала каким-либо образом инициализируем эти параметры, соблюдая ограничения, наложенные на них.

E-шаг будет устроен следующим образом: для начала определим значение $p(\mathbf{T} \mid \mathbf{X}, \Theta)$. Для этого введём скрытые переменные $\mathbf{z}_k \in \{0,1\}^l$, $z_{k1} + \ldots + z_{kl} = 1$. Они будут определять, к какой компоненте принадлежит $\mathbf{x}_k$. Тогда

$$p(\mathbf{T} \mid \mathbf{X}, \Theta) = \gamma(z_{ij}) = \frac{w_j \mathcal{N}(\mathbf{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\sum_{k=1}^{l} w_k \mathcal{N}(\mathbf{x}_k \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}.$$

Далее, выпишем $\mathsf{E}_{\mathbf{T}|\mathbf{X},\Theta_n}[\log p(\mathbf{X}, \mathbf{T} \mid \Theta)]$:

$$\begin{aligned}
\mathsf{E}_{\mathbf{T}|\mathbf{X},\Theta_n}[\log p(\mathbf{X}, \mathbf{T} \mid \Theta)] &= \sum_{\mathbf{T}} p(\mathbf{T} \mid \mathbf{X}, \Theta_n) \log p(\mathbf{X}, \mathbf{T} \mid \Theta) = \\
&= \sum_{i=1}^{n} \sum_{j=1}^{l} \gamma(z_{ij}) \log(w_j \mathcal{N}(\mathbf{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)) = \\
&= \sum_{i=1}^{n} \sum_{j=1}^{l} \gamma(z_{ij}) \left(\log w_j + \log \mathcal{N}(\mathbf{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)\right)
\end{aligned}$$

Теперь начинаем оптимизировать. Покажем вывод формулы для w_i. Воспользуемся методом Лагранжа и выпишем лагранжиан:

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^{n} \sum_{j=1}^{l} \gamma(z_{ij}) \log w_j + \lambda \left(\sum_{k=1}^{l} w_k - 1 \right).$$

Далее, дифференцируем его по w_j и получаем оптимальное значение:

$$\sum_{i=1}^{n} \frac{\gamma(z_{ij})}{w_j} + \lambda = 0 \implies w_j = -\frac{1}{\lambda} \sum_{i=1}^{n} \gamma(z_{ij})$$

Теперь воспользуемся свойствами w_j и $\gamma(z_{ij})$:

$$\sum_{j=1}^{l} w_j = -\frac{1}{\lambda} \sum_{i=1}^{n} \sum_{j=1}^{l} \gamma(z_{ij}) = -\frac{n}{\lambda} = 1 \implies \lambda = -n \implies w_j = \frac{1}{n} \sum_{i=1}^{n} \gamma(z_{ij}).$$

Без доказательства выпишем формулы для $\boldsymbol{\mu}_k$ и $\boldsymbol{\Sigma}_k$:

$$N_j = \sum_{i=1}^{n} \gamma(z_{ij}),$$
$$\boldsymbol{\mu}_j = \frac{1}{N_j} \sum_{i=1}^{n} \gamma(z_{ij}) \boldsymbol{x}_i,$$
$$\boldsymbol{\Sigma}_j = \frac{1}{N_j} \sum_{i=1}^{n} \gamma(z_{ij}) (\boldsymbol{x}_i - \boldsymbol{\mu}_j)(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^\intercal.$$

Теперь возвращаемся к E-шагу и повторяем процедуру до тех пор, пока алгоритм не сойдётся.

У EM-алгоритма много плюсов, но есть и недостатки. Например, он чувствителен к начальным приближениям — в зависимости от их значений он может сходиться к разным точкам. Далее, он находит локальный экстремум. Другими словами, алгоритм может сойтись к точке, крайне далёкой от глобального максимума — к седловой точке или, более того, к локальному минимуму. Впрочем, такое происходит не слишком часто. Стоит заметить, что в задаче разделения смеси EM-алгоритм не может определить количество компонентов смеси l — оно является *структурным параметром*.

6 Сегментация временных рядов

Задачи о сегментации временных рядов вполне распространены на практике и с ними связаны многие методы решения: например, расчёт агрегированных показателей, разбиение на равные сегменты и принцип скользящего окна. Однако они не универсальны.

Иногда на практике попадаются временные ряды, устроенные следующим образом: они весьма естественно разбиваются на достаточно однородные сегменты. Внутри них происходят колебания вокруг какого-то среднего, а после этого происходит резкий скачок и сигнал стабилизируется на другом уровне. Сами же скачки происходят неравномерно. Такая структура не позволяет эффективно использовать классические методы решения. Но при исследовании таких рядов возникла задача применимого на практике алгоритма, обеспечивающего выделение достаточно однородных фрагментов, удобных для последующей обработки.

Вообще, алгоритмы сегментации можно разделить на две большие группы: последовательные (on-line) и апостериорные (off-line). Но большинство апостериорных алгоритмов либо разбивает на два сегмента, либо использует итерационные вычислительные схемы, где одна итерация имеет сложность порядка $O(N^2)$, где N — длина известной реализации временного ряда. Было необходимо разработать апостериорный алгоритм, который, с одной стороны, мог разбивать временной ряд на произвольное число сегментов, а, с другой стороны, имел адекватную временную сложность.

Бурнаев и Меньшиков разработали такой апостериорный алгоритм, основанный на скрытых марковских моделях. В нём разбиение получается с помощью метода максимального правдоподобия. Такое разбиение является «оптимальным» в том смысле, что с точки зрения теории вероятностей ему соответствует локальный максимум функции правдоподобия, а с чисто вычислительной стороны оно минимизирует среднеквадратичное отклонение ряда от его средних значений, посчитанным по соответствующим сегментам однородности.

Ближе к делу. Переформулируем задачу сегментации временного ряда в терминах задачи максимизации функции правдоподобия некоторой скрытой марковской модели.

Пусть $\mathbf{X}^N = (x_1, \ldots, x_N)$ — известная реализация временного ряда. Будем говорить, что $\mathbf{X}^N$ является реализацией наблюдаемой части $\mathbf{X} = (X_n)_{n\geqslant 1}$ скрытой марковской модели $(\mathbf{S}, \mathbf{X}) = (S_n, X_n)_{n\geqslant 1}$, где $\mathbf{S} = (S_n)_{n\geqslant 1}$ — это марковская цепь с M состояниями, то есть $S_i \in \{1, 2, \ldots, M\}$ для всех i. По умолчанию будем считать, что начальное состояние марковской цепи $S_0 = 1$ почти наверное. Далее, введём переходную матрицу $\mathbf{P}^M = (p_{ij}) \in \mathrm{M}_M(\mathbb{R})$, для которой $p_{ij} = 0$ при $|i - j| > 1$ и $p_{MM} = 1$. Тогда по теореме **??**

$$\mathsf{P}(S_1 = s_1, \ldots, S_N = s_N) = \prod_{k=1}^{N} p_{s_{k-1}s_k}, \quad s_0 = 1.$$

Параметрами марковской цепи $\mathbf{S}$ являются количество состояний M и переходная матрица $\mathbf{P}^M$.

Теперь посмотрим на $\mathbf{X}$. Будем считать, что это последовательность условно независимых (при фиксированных значениях процесса $\mathbf{S}$) случайных величин с распределением $\mathcal{N}(\mu_{\mathbf{S}}, \sigma^2)$, то есть

$$\mathsf{P}(X_1 \leqslant x_1, \ldots, X_n \leqslant x_n \mid S_1 = s_1, \ldots, S_n = s_n) = \\ = \frac{1}{(2\pi\sigma^2)^{n/2}} \int\limits_{-\infty}^{x_1} \cdots \int\limits_{-\infty}^{x_n} \exp\left\{ -\sum_{k=1}^{n} \frac{(y_k - \mu_{s_k})^2}{2\sigma^2} \right\} dy_1 \ldots dy_n.$$

Таким образом, параметрами процесса $(\mathbf{S}, \mathbf{X})$ являются вектор средних значений $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_M)$, дисперсия σ^2 и матрица переходных вероятностей $\mathbf{P}^M$.

Введём функцию $J(\mathbf{z})$, равную количеству компонент с разными значениями в произвольном векторе $\mathbf{z} \in \mathbb{R}^N$. Далее, пусть $\mathbf{S}_N = (s_1, \ldots, s_N)$ — наблюдаемая реализация значений марковской цепи, соответсвующая реализации наблюдаемой части $\mathbf{X}^N = (x_1, \ldots, x_N)$ скрытой марковской модели. Теперь введём обозначение $M_S = J(\mathbf{S}^N)$. Вектором координат сегментов однородности временного ряда будем называть вектор $\mathbf{n}_S = (n_0, n_1, \ldots, n_{M_S})$, $n_0 = 0$, $n_{M_s} = N$, для координат которого выполнено неравенство $s_{n_i} \neq s_{n_i+1}$.

Достаточно очевидно, что $M_S \leqslant M$ и каждому сегменту однородности $[n_{i-1}, n_i]$ соответствует состояние i марковской цепи $\mathbf{S}$ и параметр μ_i распределений процесса $\mathbf{X}$. Таким образом, оценив по реализации $\mathbf{X}^n$ процесса $\mathbf{X}$ реализацию $\mathbf{S}_N$ процесса $\mathbf{S}$, можно получить оценку вектора $\mathbf{n}_S$ и, тем самым, сегментировать сигнал на участки однородности (в смысле постоянства среднего значения).

Будем оценивать $\mathbf{S}_N$ с помощью максимизирования правдоподобия при фиксированном $\mathbf{X}^N$. Обозначим условную функцию правдоподобия $\mathbf{S}_N$ через $\mathcal{L}(\mathbf{S}_N \mid \mathbf{X}^N; M, \mathbf{P}^M, \boldsymbol{\mu}, \sigma)$. Тогда

$$\mathbf{S}_N^{\mathrm{opt}} = \arg \max_{\mathbf{S}_N \in \{1,\ldots,M\}^N} \mathcal{L}(\mathbf{S}_N \mid \mathbf{X}^N; M, \mathbf{P}^M, \boldsymbol{\mu}, \sigma).$$

С помощью формулы Байеса можно получить связь между $\mathcal{L}(\mathbf{S}_N \mid \mathbf{X}^N; M, \mathbf{P}^M, \boldsymbol{\mu}, \sigma)$ и $\mathcal{L}(\mathbf{S}_N, \mathbf{X}^N; M, \mathbf{P}^M, \boldsymbol{\mu}, \sigma)$, то есть между условной и совместной функциями правдоподобия:

$$\mathcal{L}(\mathbf{S}_N \mid \mathbf{X}^N; M, \mathbf{P}^M, \boldsymbol{\mu}, \sigma) = \frac{\mathcal{L}(\mathbf{S}_N, \mathbf{X}^N; M, \mathbf{P}^M, \boldsymbol{\mu}, \sigma)}{\mathcal{L}(\mathbf{X}^N; M, \mathbf{P}^M, \boldsymbol{\mu}, \sigma)}.$$

Как известно, $\mathcal{L}(\mathbf{X}^N; M, \mathbf{P}^M, \boldsymbol{\mu}, \sigma)$ есть ни что иное, как безусловная плотность распределения случайного вектора $\mathbf{X}^N$. Следовательно, она не зависит от $\mathbf{S}^N$ и задача равносильна максимизации $\mathcal{L}(\mathbf{S}_N, \mathbf{X}^N; M, \mathbf{P}^M, \boldsymbol{\mu}, \sigma)$ по $\mathbf{S}^N$, где

$$\mathcal{L}(\mathbf{S}_N, \mathbf{X}^N; M, \mathbf{P}^M, \boldsymbol{\mu}, \sigma) = \frac{1}{(2\pi\sigma^2)^{n/2}} \prod_{k=1}^{N} p_{s_{k-1}s_k} \exp\left\{ -\frac{(x_k - \mu_{s_k})^2}{2\sigma^2} \right\}.$$

Приступим к описанию алгоритма. Но для начала сделаем одно уточнение. Будем считать, что на матрицу $\mathbf{P}^M$ наложены следующие условия: $p_{ii} = p$, $p_{i,i+1} = 1 - p$ для какого-то $p \in (0,1)$ при $i \in \{1, 2, \ldots, M-1\}$ и $p_{MM} = 1$. В дальнейшем в аргументах функции правдоподобия вместо матрицы $\mathbf{P}^M$ будем писать параметр p.

Теперь приступаем к описанию. Начнём с входных и выходных данных.

- На вход алгоритму подаются $\mathbf{X}^N = (x_1, \ldots, x_N)$ — известная реализация ряда, $M \leqslant N/2$ — оценка сверху количества сегментов и ε — пороговое значение.
- Результатом работы алгоритма являются оценки параметров p^{opt},$\boldsymbol{\mu}^{\text{opt}}$, σ^{opt}, $\mathbf{S}_N^{\text{opt}}$, $\mathbf{n}_S^{\text{opt}}$ и $M_S^{\text{opt}} = J(\mathbf{S}_N^{\text{opt}})$. При этом вполне себе возможна ситуация, когда $M_S^{\text{opt}} < M$.

Инициализация алгоритма состоит в следующем:

- Начнём с инициализации параметра p: $p^{(0)} = (N - M)/N$.
- Далее, $\mathbf{S}_N^{(0)} = (s_1^{(0)}, \ldots, s_N^{(0)})$ генерируется случайным образом из чисел множества $\{1, 2, \ldots, M\}$ с соблюдением следующих правил:

$$J(\mathbf{S}_N^{(0)}) = M, \quad s_1^{(0)} \leqslant s_2^{(0)} \leqslant \ldots s_N^{(0)}.$$

- Сразу же инициализируем оптимальную оценку стандартного отклонения:

$$\sigma^{\text{opt}} = \sqrt{\frac{1}{N-1}\sum_{k=1}^{N}(x_k - \overline{x})^2}, \text{ где } \overline{x} = \frac{1}{N}\sum_{k=1}^{N} x_k.$$

Теперь оговорим условие остановки алгоритма. Пусть m — номер текущей итерации. Алгоритм остановит работу в том случае, если изменение функции правдоподобия стало достаточно малым:

$$|\mathcal{L}(\mathbf{S}_N^{(m)}, \mathbf{X}^N; M_S^{(m)}, p^{(m)}, \boldsymbol{\mu}^{(m)}, \sigma^{\text{opt}}) - \mathcal{L}(\mathbf{S}_N^{(m-1)}, \mathbf{X}^N; M_S^{(m-1)}, p^{(m-1)}, \boldsymbol{\mu}^{(m-1)}, \sigma^{\text{opt}})| < \varepsilon.$$

Теперь опишем, что происходит на m-й итерации алгоритма:

1. По значению $\mathbf{S}_N^{(m)}$ строится вектор $\mathbf{n}_S^{(m)}$ и определяется $M_S^{(m)} = J(\mathbf{S}_N^{(m)})$.
2. По вектору $\mathbf{n}_S^{(m)}$ строится вектор матожиданий $\boldsymbol{\mu}^{(m)} \in \mathbb{R}^{M_S^{(m)}}$ по следующему правилу:

$$\mu_i^{(m)} = \frac{1}{n_i - n_{i-1}} \sum_{k=n_{i-1}+1}^{n_i} x_k, \quad i \in \{1, 2, \ldots, M_S^{(m)}\}.$$

3. Ищется вероятность $p^{(m)} = (N - M_S^{(m)})/N$.
4. Оценивается значение функции правдоподобия $\mathcal{L}(\mathbf{S}_N^{(m)}, \mathbf{X}^N; M^{(m)}, p^{(m)}, \boldsymbol{\mu}^{(m)}, \sigma^{\text{opt}})$ и проверяется, выполнено ли условие остановки алгоритма. Если оно подтверждено, то мы получили ответ: $p^{\text{opt}} = p^{(m)}$,$\boldsymbol{\mu}^{\text{opt}} = \boldsymbol{\mu}^{(m)}$, $\mathbf{S}_N^{\text{opt}} = \mathbf{S}_N^{(m)}$, $\mathbf{n}_S^{\text{opt}} = \mathbf{n}_S^{(m)}$ и $M_S^{\text{opt}} = M_S^{(m)} = J(\mathbf{S}_N^{(m)})$. Иначе же переходим к следующему шагу.
5. С помощью алгоритма Витерби инициализируем $\mathbf{S}_N^{(m+1)}$:

$$\mathbf{S}_N^{(m+1)} = \arg \max_{\mathbf{S}_N \in \{1,\ldots,M_S^{(m)}\}^N} \mathcal{L}(\mathbf{S}_N, \mathbf{X}^N; M_S^{(m)}, p^{(m)}, \boldsymbol{\mu}^{(m)}, \sigma^{\text{opt}}).$$

После этого m увеличивается на единицу и запускается следующая итерация.

Описанный выше алгоритм, по сути, можно отнести к классу EM-алгоритмов, так как сначала алгоритм сводится к подсчёту функции правдоподобия и её максимизации. На практике обычно смотрят не на саму функцию правдоподобия, а на её логарифм.

Теперь докажем одно простое утверждение, которое доказывает корректность алгоритма.

Теорема 2. *Описанный выше алгоритм сходится.*

Доказательство. Введём следующую функцию:

$$F(\mathbf{S}_N, \mathbf{X}^N; M, p, \boldsymbol{\mu}, \sigma) \equiv -\ln \mathcal{L}(\mathbf{S}_N, \mathbf{X}^N; M, p, \boldsymbol{\mu}, \sigma).$$

Несложно получить, что

$$F(\mathbf{S}_N, \mathbf{X}^N; M, p, \boldsymbol{\mu}, \sigma) = \sum_{k=1}^{N} \frac{(x_k - \mu_{s_k})^2}{2\sigma^2} - \sum_{k=1}^{N} \ln p_{s_{k-1}s_k} + N \ln(\sigma\sqrt{2\pi}).$$

Рассмотрим сумму логарифмов отдельно. Мы можем сказать (по построению $\mathbf{S}_N$), что $p_{s_{k-1}s_k}$ равно либо p, либо $1-p$. Если $p_{s_{k-1}s_k} = 1-p$, то $s_k = s_{k-1} + 1$. Если же $p_{s_{k-1}s_k} = p$, то $s_k = s_{k-1}$. Следовательно,

$$\sum_{k=1}^{N} \ln p_{s_{k-1}s_k} = J(\mathbf{S}_N)\ln(1-p) + (N - J(\mathbf{S}_N))\ln p = J(\mathbf{S}_N)\ln\frac{1-p}{p} + N\ln p.$$

Отсюда мы получаем, что верно следующее представление:

$$F(\mathbf{S}_N, \mathbf{X}^N; M, p, \boldsymbol{\mu}, \sigma) = \frac{1}{2\sigma^2} F_1(\mathbf{S}_N, \mathbf{X}^N; M, \boldsymbol{\mu}) + F_2(\mathbf{S}_N; M, p) + N\ln(\sigma\sqrt{2\pi}),$$

где функции F_1 и F_2 опредеяются следующим образом:

$$F_1(\mathbf{S}_N, \mathbf{X}^N; M, \boldsymbol{\mu}) = \sum_{k=1}^{N} (x_k - \mu_{s_k})^2, \quad F_2(\mathbf{S}_N; M, p) = J(\mathbf{S}_N)\ln\frac{p}{1-p} - N\ln p.$$

Теперь заметим, что выполнены три неравенства:

- Для всех $\boldsymbol{\mu} \in \mathbb{R}^{M_S^{(m)}}$, где $M_S^{(m)} = J(\mathbf{S}_N^{(m)})$, выполнено следующее неравенство:

$$F_1(\mathbf{S}_N^{(m)}, \mathbf{X}^N; M_S^{(m)}, \boldsymbol{\mu}) \geqslant F_1(\mathbf{S}_N^{(m)}, \mathbf{X}^N; M_S^{(m)}, \boldsymbol{\mu}^{(m)}).$$

- Для всех $p \in (0,1)$ выполнено, что

$$F_2(\mathbf{S}_N^{(m)}; M_S^{(m)}, p) \geqslant F_2(\mathbf{S}_N^{(m)}; M_S^{(m)}, p^{(m)}).$$

  Действительно, возьмём производную по p и приравняем её к нулю:

$$J(\mathbf{S}_N^{(m)})\frac{1-p}{p}\frac{1-p+p}{(1-p)^2} - \frac{N}{p} = 0.$$

  Отсюда получаем, что

$$\frac{J(\mathbf{S}_N^{(m)})}{1-p} = N \implies p = 1 - \frac{J(\mathbf{S}_N^{(m)})}{N} = \frac{N - J(\mathbf{S}_N^{(m)})}{N} \equiv p^{(m)}.$$

  Так как при увеличении p знак производной меняется с отрицательного на положительный, то это действительно минимум.

- Так как алгоритм Витерби позволяет получить глобальный максимум функции правдоподобия, то для любого $\mathbf{S}_N \in \{1, 2, \ldots, M_S^{(m)}\}^N$

$$F(\mathbf{S}_N, \mathbf{X}^N; M_S^{(m)}, p, \boldsymbol{\mu}^{(m)}, \sigma) \geqslant F(\mathbf{S}_N^{(m)}, \mathbf{X}^N; M_S^{(m)}, p, \boldsymbol{\mu}^{(m)}, \sigma).$$

Отсюда получаем следующую цепочку равенств:

$$F(\mathbf{S}_N^{(m-1)}, \mathbf{X}^N; M_S^{(m-1)}, p^{(m-1)}, \boldsymbol{\mu}^{(m-1)}, \sigma) \geqslant F(\mathbf{S}_N^{(m)}, \mathbf{X}^N; M_S^{(m-1)}, p^{(m-1)}, \boldsymbol{\mu}^{(m-1)}, \sigma) \geqslant$$
$$\geqslant F(\mathbf{S}_N^{(m)}, \mathbf{X}^N; M_S^{(m)}, p^{(m)}, \boldsymbol{\mu}^{(m)}, \sigma) \geqslant F(\mathbf{S}_N^{(m+1)}, \mathbf{X}^N; M_S^{(m)}, p^{(m)}, \boldsymbol{\mu}^{(m)}, \sigma).$$

Отсюда получаем, что каждая итерация уменьшает $F(\mathbf{S}_N^{(m)}, \mathbf{X}^N; M_S^{(m)}, p^{(m)}, \boldsymbol{\mu}^{(m)}, \sigma)$. Так как это значение ограничено снизу нулём, то существует предел и алгоритм сходится. □

Теперь выпишем несколько свойств этого алгоритма.

- Каждая итерация имеет сложность порядка $O(NM^2)$ (такая оценка возникает из-за алгоритма Витерби). Эксперименты показывают, что он сходится за 10-15 итераций, что характерно для алгоритмов типа EM.

- Как мы показали выше, минимум $F_2(\mathbf{S}_N; M, p)$ достигается при $p^{\text{opt}} = (N - J(\mathbf{S}_N))/N$ и равен

$$F_2(\mathbf{S}_N; M, p^{(m)}) = J(\mathbf{S}_N) \ln\left(\frac{N - J(\mathbf{S}_N)}{J(\mathbf{S}_N)}\right) - N \ln\left(\frac{N - J(\mathbf{S}_N)}{N}\right) =$$
$$= J(\mathbf{S}_N) \ln\left(\frac{N}{J(\mathbf{S}_N)} - 1\right) - N \ln\left(1 - \frac{J(\mathbf{S}_N)}{N}\right).$$

  Оказывается, что $F_2(\mathbf{S}_N; M, p^{\text{opt}})$ отвечает за регуляризацию, поскольку при увеличении $J(\mathbf{S}_N)$ это слагаемое возрастает, в то время как $F_1(\mathbf{S}_N, \mathbf{X}^N; M, \boldsymbol{\mu})$ убывает (и наоборот). Следовательно, при использовании алгоритма Витерби при минимизации $F(\mathbf{S}_N, \mathbf{X}^N; M, p, \boldsymbol{\mu}, \sigma)$ по $\mathbf{S}_N \in \{1, 2, \ldots, M\}^N$ при фиксированных значениях M и $\boldsymbol{\mu}$ будет так же выбрано оптимальное число сегментов разбиения, равное $M_S = J(\mathbf{S}_N) \leqslant M$. Вообще говоря, $M \geqslant M_S^{(0)} \geqslant M_S^{(1)} \geqslant \ldots$, причём процесс понижения значения $J(\mathbf{S}_N)$ со временем стабилизируется, так как иначе значение F начнёт возрастать.

- Далее, после стабилизации процесса понижения значения $J(\mathbf{S}_N)$, значение F понижается дальше за счёт «подстройки» значений $\mathbf{n}_S$ (и, следовательно, значений, $\boldsymbol{\mu}$). Описанный нами алгоритм, по сути, ищет такое разбиение, при котором приближение временного ряда его средними значениями по соответствующим сегментам будет оптимальным в смысле среднеквадратичного отклонения.

- Практика показала следующее: результат работы алгоритма слабо зависит от $p^{(0)}$ и $\mathbf{S}_N^{(0)}$, но сильно зависит от M. Если взять $M \ll N$, то с большой вероятностью $M_S^{\text{opt}} = M$. В противном случае $M_S^{\text{opt}} < M$. Более того, если запустить алгоритм с двумя разными параметрами M_1 и M_2, то полученные значения $M_S^{\text{opt},1}$ и $M_S^{\text{opt},2}$ не обязательно равны. Это вытекает из того факта, что оптимизируемая функция многоэкстремальна.