2 EM-алгоритм. Численный поиск ОМП

2.1 Отсутствие ОМП для смеси нормальных распределений

Вычисление ОМП может быть достаточно затруднительным или вообще невозможным. Рассмотрим, например, следующую задачу:

Пример 1. Пусть X_i являются смесью двух нормальных распределений, то есть каждое наблюдение с вероятностью p получается из распределения $\mathcal{N}(\mu_1, \sigma_1^2)$, а иначе из $\mathcal{N}(\mu_2, \sigma_2^2)$. Тогда правдоподобие имеет вид

$$L(x_1, \dots, x_n; p, \mu_1, \mu_2, \sigma_1, \sigma_2) = f_{p,\mu_1,\mu_2,\sigma_1,\sigma_2}(x_1) \cdots f_{p,\mu_1,\mu_2,\sigma_1,\sigma_2}(x_1) = \prod_{i=1}^{n} \left(\frac{p}{\sqrt{2\pi}\sigma_1} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} + \frac{1-p}{\sqrt{2\pi}\sigma_2} e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}} \right).$$

Эта функция выглядит крайне громоздкой и максимизация ее выглядит сложной задачей. В действительности все еще хуже, если положить $p = 1/2$, $\mu_1 = x_1$, $\mu_2 = 0$, $\sigma_2 = 1$ и устремить $\sigma_1 \to 0$, то правдоподобие можно сделать сколь угодно большим.

Вопрос 1. Докажите этот факт.

Однако, даже если $\sigma_1 = 1$, $\sigma_2 = 1$, $p = 1/2$ и правдоподобие ограничено, найти его максимум будет затруднительно.

2.2 EM-алгоритм

Может оказаться, что глобального максимума функции правдоподобия не существует, но есть локальные, причем какие-то из них могут давать хорошую оценку (собственно, в самой асимптотической теории для ОМП рассматриваются именно решения уравнения правдоподобия — уравнения на производную правдоподобия, задающего локальные максимумы).

Предложим метод, который может довольно эффективно искать локальные максимумы в такого рода сложных задачах, называемый EM-алгоритмом (по двум шагам алгоритма — Expectation и Maximization).

Пусть мы хотим максимизировать правдоподобие $L(x_1, \dots, x_n; \vec{\theta})$. Предположим, что наша задача значительно упростилась бы, если бы вместе с выборкой $x_1, \dots, x_n$ мы знали некоторые скрытые наблюдения $y_1, \dots, y_m$. Так для задачи из прошлого примера мы легко решили бы задачу, если бы мы знали $y_1, \dots, y_n \in \{1, 2\}$, отвечающие за то к какому из нормальных распределений относится каждый элемент.

Тогда EM-алгоритм предлагает следующую программу действий:

1. Выберем начальное значение параметров $\vec{\theta}^0 = (\theta_1^0, \dots, \theta_k^0)$ в нашей модели.
   На $j+1$-ом шаге проделаем следующее:
2. Expectation. Вычислим
$$J(\vec{\theta}|\vec{\theta^j}) = \mathbf{E}_{\vec{\theta}^j} \left(\ln \frac{L(X_1, \dots, X_n, Y_1, \dots, Y_m; \vec{\theta})}{L(X_1, \dots, X_n, Y_1, \dots, Y_m; \vec{\theta}^j)} \middle| X_1 = x_1, \dots, X_n = x_n \right).$$
3. Maximization. Выберем $\vec{\theta}_{j+1}$ так, что $J(\vec{\theta}^{j+1}|\vec{\theta}^j)$ максимально.

Можно доказать, что на каждом шаге функция правдоподобия не уменьшается.

Бывает удобно слегка переформулировать процедуру:

1. Вычислим распределение $y_1, \dots, y_m$ в дискретном случае
$$\mathbf{P}_{\vec{\theta}^j}(Y_1 = y_1, \dots, Y_m = y_m | X_1 = x_1, \dots, X_n = x_n) = \frac{L(x_1, \dots, x_n, y_1, \dots, y_m; \vec{\theta}^j)}{\mathbf{E}_{\vec{\theta}^j} L(x_1, \dots, x_n, Y_1, \dots, Y_m; \vec{\theta}^j)}$$

или плотность $Y_1, \dots, Y_m$ в непрерывном случае

$$f_{\vec{\theta}^j}(y_1, \dots, y_m | X_1 = x_1, \dots, X_n = x_n) = \frac{L(x_1, \dots, x_n, y_1, \dots, y_m; \vec{\theta}^j)}{\mathbf{E}_{\vec{\theta}^j} L(x_1, \dots, x_n, Y_1, \dots, Y_m; \vec{\theta}^j)}.$$

Такое распределение называется апостериорным для $Y_1, \dots, Y_m$ при условии $X_1, \dots, X_n$.

2. Максимизируем по θ

$$J(\vec{\theta}|\vec{\theta}^j) = \sum_{y_1, \dots, y_m} \mathbf{P}_{\vec{\theta}^j}(Y_1 = y_1, \dots, Y_m = y_m | X_1 = x_1, \dots, X_n = x_n) \ln L(x_1, \dots, x_n, y_1, \dots, y_m; \vec{\theta})$$

в дискретном или

$$J(\vec{\theta}|\vec{\theta}^j) = \int_{\mathbb{R}^m} f_{\vec{\theta}^j}(y_1, \dots, y_m | X_1 = x_1, \dots, X_n = x_n) \ln L(x_1, \dots, x_n, y_1, \dots, y_m; \vec{\theta}) dy_1 \dots dy_m$$

в абсолютно-непрерывном. Иначе говоря,

$$J(\vec{\theta}|\vec{\theta}^j) = \mathbf{E} \ln L(x_1, \dots, x_n, Y_1, \dots, Y_n; \theta),$$

взятое по апостериорному распределению $\widetilde{\mathbf{P}}(A) = \mathbf{P}_{\vec{\theta}^j}((Y_1, \dots, Y_n) \in A | X_1 = x_1, \dots, X_n = x_n)$.

Во второй форме видно, чем EM-алгоритм облегчает нашу задачу. Максимизация правдоподобия соответствует шагам 1'), 2'), где на первом шаге используется не известное $\vec{\theta}^j$, а неизвестное $\vec{\theta}$, по которому и ведется максимизация. Мы же упрощаем задачу, подставляя на первом шаге текущую оценку параметра $\vec{\theta}$ и улучшая ее последовательными итерациями.

Улучшая полученные оценки $\vec{\theta}^j$, мы постепенно будем приближаться к точке локального (!) максимума правдоподобия, либо, если такой точки нет, уходить в бесконечность. При этом даже в условиях прошлого примера мы вполне можем получить какую-то оценку, поскольку можем попасть в один из локальных экстремумов. С другой стороны, мы совершенно не гарантируем, что этот экстремум будет близок к оцениваемому параметру, да и то, какой именно получится экстремум, зависит от начальной оценки для наших параметров.

2.3 EM-алгоритм для смеси нормальных распределений

Давайте применим полученный метод к уже упоминавшейся задаче о смеси нормальных:
Пусть $X_1, \dots, X_n$ — наши величины, полученные из $\mathcal{N}(\mu_1, 1)$ или $\mathcal{N}(\mu_2, 1)$ с вероятностью $p = 0.5$, $\vec{\theta} = (\mu_1, \mu_2)$, $Y_1, \dots, Y_n$ — величины, отвечающие за то, из какого распределения получены величины X_i. Тогда

$$L(x_1, \dots, x_n, y_1, \dots, y_n; \mu_1, \mu_2) = \frac{1}{2^n} \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\sum_{i=1}^{n}(x_i - \mu_{y_i})^2}.$$

Воспользуемся формулой

$$J(\vec{\theta}|\vec{\theta}^j) = \mathbf{E}_{\vec{\theta}} \ln L(x_1, \dots, x_n, Y_1, \dots, Y_m) = -n \ln 2 - \frac{n}{2} \ln(2\pi) - \frac{1}{2} \sum_{i=1}^{n} \mathbf{E}_{\widetilde{\mathbf{P}}}(x_i - \mu_{Y_i})^2,$$

где математическое ожидание, как уже упоминалось, берется по мере $\widetilde{\mathbf{P}}(A)$, заданной распределением Y при условии X, то есть апостериорным распределением Y при условии X. Давайте найдем его по формуле Байеса

$$p_i^j = \widetilde{\mathbf{P}}(Y_i = 1) = \mathbf{P}_{\vec{\theta}^j}(Y_i = 1 | X_1 = x_1, \dots, X_n = x_n) =$$
$$\frac{\mathbf{P}(Y_i = 1) f_{X_1}(x_1) \cdots f_{X_{i-1}}(x_{i-1}) f_{X_i}(x_i | Y_i = 1) f_{X_{i+1}}(x_{i+1}) \cdots f_{X_n}(x_n)}{\prod_{j \neq i} f_{X_i}(x_i) \left(\mathbf{P}_{\vec{\theta}^j}(Y_i = 1) f_{X_i}(x_i | Y_i = 1) + \mathbf{P}_{\vec{\theta}^j}(Y_i = 0) f_{X_i}(x_i | Y_i = 0) \right)} =$$
$$\frac{\exp(-\frac{1}{2}(x_i - \mu_1^j)^2)}{\exp(-\frac{1}{2}(x_i - \mu_1^j)^2) + \exp(-\frac{1}{2}(x_i - \mu_0^j)^2)}. \quad (1)$$

Значит

$$\mathbf{E}_{\widetilde{\mathbf{P}}}(x-\mu_{Y_i})^2 = (x-\mu_1)^2\widetilde{\mathbf{P}}(Y_i=1) + (x-\mu_2)^2\widetilde{\mathbf{P}}(Y_i=2).$$

Отсюда максимизация $J(\vec{\theta}|\vec{\theta^j})$ сводится к минимизации

$$\sum_{i=1}^{n}(p_i^j(x_i-\mu_1)^2 + (1-p_i^j)(x_i-\mu_2)^2).$$

Дифференцируя, получаем

$$\mu_1^{j+1} = \frac{\sum_{i=1}^n p_i x_i}{\sum_{i=1}^n p_i}, \quad \mu_2^{j+1} = \frac{\sum_{i=1}^n (1-p_i) x_i}{\sum_{i=1}^n (1-p_i)}. \tag{2}$$

Итак, наш алгоритм готов:

1. Находим p_i^j на основе μ_i^j по формуле (1).
2. Находим μ_i^{j+1} по формуле (2).
3. Переходим на следующий шаг и повторяем процедуру.

Задача 1. Протестировать метод для $\mu_1 = 0$, $\mu_2 = 1$.

EM-алгоритм для нормальных данных реализован в R в пакетах EMCluster или mclust, в Python в модуле sklearn в пакете mixture. Примеры программ можно найти в разделе практикума.

2.4 EM-алгоритм для неполных данных

EM-алгоритм также крайне удобен, если у нас неполные данные.

Пусть $Z_1, \ldots, Z_n$ из полиномиального распределения. Напомню, что полиномиальное распределение соответствует эксперименту, в котором проводится m испытаний, в каждом из которых n возможных итогов с вероятностями $\tau_1, \ldots, \tau_n$, $Z_1, \ldots, Z_n$ при этом будет означать количество исходов каждого из видов.

Представим себе, что $\beta \in \mathbb{R}^+$, $m \in \mathbb{N}$, $\tau_i \in [0,1]$: $\tau_1 + \cdots + \tau_n = 1$, X_i — пуассоновские с параметром $m\beta\tau_i$, а $(Z_1, \ldots, Z_n)$ — полиномиальный вектор с вероятностями $\tau_1, \ldots, \tau_n$ и числом испытаний m. При этом наблюдение Z_1 отсутствует среди наших наблюдений, параметры β, τ_i нам неизвестны. Мы хотим оценить β, τ_i на основе данных $(X_1, \ldots, X_n)$, $(Z_2, \ldots, Z_n)$. Правдоподобие при известном z_1 имело бы вид

$$L(x_1, \ldots, x_n, z_1, z_2, \ldots, z_n; \beta, \tau_i) = \frac{(z_1 + \cdots + z_n)!}{z_1! \cdots z_n!} \tau_1^{z_1} \cdots \tau_n^{z_n} \prod_{i=1}^{n} e^{-x_i} \frac{(m\beta\tau_i)^{x_i}}{x_i!} e^{-m\beta\tau_i}$$

При неизвестном z_1 задача максимизации сложна. Возьмем в качестве Y_1 неизвестную величину Z_1. Теперь мы можем применить EM-алгоритм (опять же во второй форме — вычисляя апостериорное распределение z_1 при известных параметрах $\vec{\theta^j}$).

Задача 2. Применить EM-алгоритм к этой задаче и реализовать его на R или Python, полагая $n = 10$, $\tau_i = 1/10$,$\beta = 0.25$, $m = 60$. Сгенерировать полиномиальную выборку с равными вероятностями исходов в R можно с помощью функции sample: sample(1:n, m, replace=TRUE). К слову, можно аналогично генерировать и неравновероятную полиномиальную выборку, указывая среди параметров prob = p, где p — вектор вероятностей. В Python аналогичный трюк можно проделать, используя random.choice из numpy, где параметр p, опять-таки, отвечает вероятностям исходов.