

КЛАСТЕРИЗАЦИЯ И АЛГОРИТМ ЕМ

Сергей Николенко

СПбГУ — Санкт-Петербург

01 марта 2024 г.



Random facts:

- 1 марта 5500 г. до Рождества Христова — шестой день творения, вычисленный согласно Септуагинте византийскими теологами; это летоисчисление (Константинопольская эра) начиная с VII века стало общепринятым в Византии и во всём православном мире
- 1 марта в Болгарии — праздник Баба Марта в честь, собственно, Бабы Марты, доброй младшей сестры Большого Сечко (января) и Маленького Сечко (февраля)
- 1 марта — Международный день ООН «Ноль дискриминации», отмечающийся с 2014 года
- 1 марта 1564 г. вышла первая точно датированная русская печатная книга — «Апостол» Ивана Фёдорова
- 1 марта 1692 г. власти города Салем (Массачусетс) обвинили трёх женщин в колдовстве (и это было только начало); а 1 марта 1780 г. другой новоанглийский штат Пенсильвания стал первым штатом, запретившим рабство
- 1 марта 1869 г. Дмитрий Менделеев завершил свой «Опыт системы элементов, основанной на их атомном весе и химическом сходстве»

КЛАСТЕРИЗАЦИЯ

- *Кластеризация* — типичная задача обучения без учителя: задача классификации объектов одной природы в несколько групп так, чтобы объекты в одной группе обладали одним и тем же свойством.
- Под свойством обычно понимается близость друг к другу относительно выбранной метрики.

- Есть набор тестовых примеров $X = \{x_1, \dots, x_n\}$ и функция расстояния между примерами ρ .
- Требуется разбить X на непересекающиеся подмножества (кластеры) так, чтобы каждое подмножество состояло из похожих объектов, а объекты разных подмножеств существенно различались.

- Есть точки $x_1, x_2, \dots, x_n$ в пространстве. Нужно кластеризовать.
- Считаем каждую точку кластером. Затем ближайшие точки объединяем, далее считаем единым кластером. Затем повторяем.
- Получается дерево.

HierarchyCluster($X = \{x_1, \dots, x_n\}$)

- Инициализируем $C = X, G = X$.
- Пока в C больше одного элемента:
 - Выбираем два элемента C c_1 и c_2 , расстояние между которыми минимально.
 - Добавляем в G вершину c_1c_2 , соединяем её с вершинами c_1 и c_2 .
 - $C := C \cup \{c_1c_2\} \setminus \{c_1, c_2\}$.
- Выдаём G .

- В итоге получается дерево кластеров, из которого потом можно выбрать кластеризацию с требуемой степенью детализации (обрезать на том или ином максимальном расстоянии).
- Всё ли понятно?

- В итоге получается дерево кластеров, из которого потом можно выбрать кластеризацию с требуемой степенью детализации (обрезать на том или ином максимальном расстоянии).
- Всё ли понятно?
- Остаётся вопрос: как подсчитывать расстояние между кластерами?

- *Single-link* алгоритмы считают *минимум* из возможных расстояний между парами объектов, находящихся в кластере.
- *Complete-link* алгоритмы считают *максимум* из этих расстояний
- Какие особенности будут у single-link и complete-link алгоритмов? Чем они будут отличаться?

- Нарисуем полный граф с весами, равными расстоянию между объектами.
- Выберем некий предопределённый порог расстояния r и выбросим все рёбра длиннее r .
- Компоненты связности полученного графа — это наши кластеры.

- Минимальное остовное дерево — дерево, содержащее все вершины (связного) графа и имеющее минимальный суммарный вес своих рёбер.
- Алгоритм Краскала (Kruskal): выбираем на каждом шаге ребро с минимальным весом, если оно соединяет два дерева, добавляем, если нет, пропускаем.
- Алгоритм Борувки (Boruvka).

- Как использовать минимальное остовное дерево для кластеризации?

- Как использовать минимальное остовное дерево для кластеризации?
- Построить минимальное остовное дерево, а потом выкидывать из него рёбра максимального веса.
- Сколько рёбер выбросим, столько кластеров получим.

- Идея: кластер – это зона высокой плотности точек, отделённая от других кластеров зонами низкой плотности.
- Алгоритм: выделяем *core samples*, которые сэмплируются в зонах высокой плотности (т.е. есть по крайней мере n соседей, других точек на расстоянии $\leq \epsilon$).
- Затем последовательно объединяем *core samples*, которые оказываются соседями друг друга.
- Точки, которые не являются ничьими соседями, — это выбросы.

- Идея: строим дерево (CF-tree, от clustering feature), которое содержит краткие описания кластеров и поддерживает апдейты.
- $CF_i = \{N_i, LS_i, SS_i\}$: число точек в кластере CF_i ,
 $LS_i = \sum_{\mathbf{x} \in CF_i} x_i$ (linear sum), $SS_i = \sum_{\mathbf{x} \in CF_i} x_i^2$ (sum of squares).
- Этого достаточно для того, чтобы подсчитать разумные расстояния между кластерами.
- А также для того, чтобы слить два кластера: CF_i аддитивны.

- CF-дерево состоит из CF_i ; оно похоже на B-дерево, сбалансировано по высоте. Кластеры – листья дерева, над ними “суперкластеры”.
- Добавляем новый кластер, рекурсивно вставляя его в дерево; если от этого число элементов в листе становится слишком большим (параметр), лист разбивается на два.
- А когда дерево построено, можно запустить ещё одну кластеризацию (любым другим методом) на полученных “мини-кластерах”.

Алгоритм EM

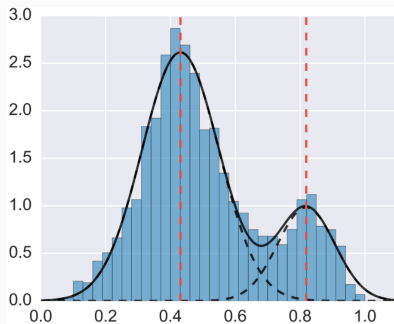
- Часто возникает ситуация, когда в имеющихся данных некоторые переменные присутствуют, а некоторые — отсутствуют.
- Даны результаты сэмплирования распределения вероятностей с несколькими параметрами, из которых известны не все.

- Эти неизвестные параметры тоже расцениваются как случайные величины.
- Задача — найти наиболее вероятную гипотезу, то есть ту гипотезу h , которая максимизирует

$$E[\ln p(D|h)].$$

ЧАСТНЫЙ СЛУЧАЙ

- Построим один из простейших примеров применения алгоритма ЕМ. Пусть случайная переменная y сэмплируется из суммы двух нормальных распределений. Дисперсии даны (одинаковые), нужно найти только средние μ_1, μ_2 .



- Какое тут правдоподобие? Как его оптимизировать?

- Нельзя понять, какие y_i были порождены каким распределением — классический пример *скрытых переменных*.
- Один тестовый пример полностью описывается как тройка $\langle y_i, z_{i1}, z_{i2} \rangle$, где $z_{ij} = 1$ iff y_i был сгенерирован j -м распределением.

- Сгенерировать какую-нибудь гипотезу $h = (\mu_1, \mu_2)$.
- Пока не дойдем до локального максимума:
 - Вычислить ожидание $E(z_{ij})$ в предположении текущей гипотезы (E -шаг).
 - Вычислить новую гипотезу $h' = (\mu'_1, \mu'_2)$, предполагая, что z_{ij} принимают значения $E(z_{ij})$ (M -шаг).

- В примере с гауссианами:

$$\begin{aligned} E(z_{ij}) &= \frac{p(y = y_i | \mu = \mu_j)}{p(y = y_i | \mu = \mu_1) + p(y = y_i | \mu = \mu_2)} = \\ &= \frac{e^{-\frac{1}{2\sigma^2}(y_i - \mu_j)^2}}{e^{-\frac{1}{2\sigma^2}(y_i - \mu_1)^2} + e^{-\frac{1}{2\sigma^2}(y_i - \mu_2)^2}}. \end{aligned}$$

- Мы подсчитываем эти ожидания, а потом подправляем гипотезу:

$$\mu_j \leftarrow \frac{1}{m} \sum_{i=1}^m E(z_{ij}) y_i.$$

- Звучит логично, но с какой стати это всё работает? Об этом потом. :)

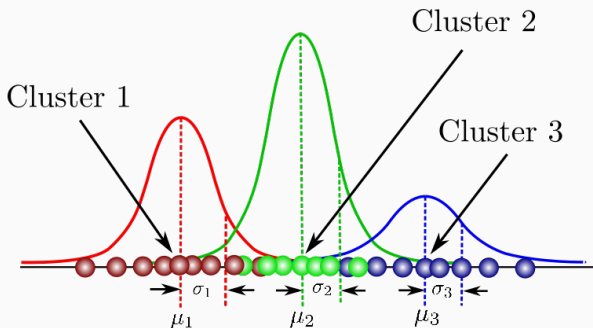
ЕМ для кластеризации

Мысли?

- Какие есть мысли о применении алгоритма EM к задачам кластеризации?

Мысли?

- Какие есть мысли о применении алгоритма EM к задачам кластеризации?
- Кластеризацию можно формализовать как всё ту же задачу разделения смеси распределений:



- Чтобы воспользоваться статистическим алгоритмом, нужно сформулировать гипотезы о распределении данных.
- *Гипотеза о природе данных*: тестовые примеры появляются случайно и независимо, согласно вероятностному распределению, равному смеси распределений кластеров

$$p(\mathbf{y}) = \sum_{c \in C} w_c p_c(\mathbf{y}), \quad \sum_{c \in C} w_c = 1,$$

где w_c — вероятность появления объектов из кластера c , p_c — плотность распределения кластера c .

- Остается вопрос: какими предположить распределения p_c ?

- Остается вопрос: какими предположить распределения p_c ?
- Часто берут сферические гауссианы, но это не слишком гибкий вариант: кластер может быть вытянут в ту или иную сторону.

- Остается вопрос: какими предположить распределения p_c ?
- Часто берут сферические гауссианы, но это не слишком гибкий вариант: кластер может быть вытянут в ту или иную сторону.
- Можно взять, например, эллиптические гауссианы.
- *Гипотеза 2*: Каждый кластер c описывается d -мерной гауссовской плотностью с центром $\mu_c = \{\mu_{c1}, \dots, \mu_{cd}\}$ и диагональной матрицей ковариаций $\Sigma_c = \text{diag}(\sigma_{c1}^2, \dots, \sigma_{cd}^2)$ (т.е. по каждой координате своя дисперсия).

ПОСТАНОВКА ЗАДАЧИ И ОБЩИЙ ВИД АЛГОРИТМА

- В этих предположениях получается в точности задача разделения смеси вероятностных распределений. Для этого и нужен EM-алгоритм.
- Каждый тестовый пример описывается своими координатами $(\mathbf{y}_1, \dots, \mathbf{y}_N)$.
- Скрытые переменные $\mathbf{z}_n$ в данном случае — это one-hot вектор $\mathbf{z}_n = (z_{nc})$ того, какому кластеру принадлежит $\mathbf{y}_n$.
- Обозначим вероятности того, что объект $\mathbf{y}_n$ принадлежит кластеру $c \in C$, через g_{nc} .

- E -шаг: по формуле Байеса вычисляются скрытые переменные g_{nc} :

- E -шаг: по формуле Байеса вычисляются скрытые переменные g_{nc} :

$$g_{nc} = \frac{w_c p_c(\mathbf{y}_n)}{\sum_{c' \in C} w_{c'} p_{c'}(\mathbf{y}_n)}.$$

- E -шаг: по формуле Байеса вычисляются скрытые переменные g_{nc} :

$$g_{nc} = \frac{w_c p_c(\mathbf{y}_n)}{\sum_{c' \in C} w_{c'} p_{c'}(\mathbf{y}_n)}.$$

- M -шаг: с использованием g_{nc} уточняются параметры кластеров $\mathbf{w}, \mu, \sigma$:

- E -шаг: по формуле Байеса вычисляются скрытые переменные g_{nc} :

$$g_{nc} = \frac{w_c p_c(\mathbf{y}_n)}{\sum_{c' \in C} w_{c'} p_{c'}(\mathbf{y}_n)}.$$

- M -шаг: с использованием g_{nc} уточняются параметры кластеров $\mathbf{w}$, μ , σ :

$$w_c = \frac{1}{N} \sum_{n=1}^N g_{nc}, \quad \mu_c = \frac{1}{nw_c} \sum_{n=1}^N g_{nc} \mathbf{y}_n,$$

- E -шаг: по формуле Байеса вычисляются скрытые переменные g_{nc} :

$$g_{nc} = \frac{w_c p_c(\mathbf{y}_n)}{\sum_{c' \in C} w_{c'} p_{c'}(\mathbf{y}_n)}.$$

- M -шаг: с использованием g_{nc} уточняются параметры кластеров $\mathbf{w}$, μ , σ :

$$w_c = \frac{1}{N} \sum_{n=1}^N g_{nc}, \quad \mu_c = \frac{1}{nw_c} \sum_{n=1}^N g_{nc} \mathbf{y}_n,$$

$$\sigma_{cj}^2 = \frac{1}{nw_c} \sum_{n=1}^N g_{nc} (y_{nj} - \mu_{cj})^2.$$

EMCluster($Y, |C|$):

- Инициализировать $|C|$ кластеров; начальное приближение:
 $w_c := 1/|C|$, $\mu_c :=$ случайный $\mathbf{y}_n$, $\sigma_{cj}^2 := \frac{1}{n|C|} \sum_{i=1}^N (y_{nj} - \mu_{cj})^2$.
- Пока принадлежность кластерам не перестанет изменяться:
 - E -шаг: $g_{nc} := \frac{w_c p_c(\mathbf{y}_n)}{\sum_{c' \in C} w_{c'} p_{c'}(\mathbf{y}_n)}$.
 - M -шаг: $w_c = \frac{1}{N} \sum_{i=1}^N g_{nc}$, $\mu_{cj} = \frac{1}{nw_c} \sum_{i=1}^N g_{nc} y_{nj}$,

$$\sigma_{cj}^2 = \frac{1}{nw_c} \sum_{i=1}^N g_{nc} (y_{nj} - \mu_{cj})^2.$$

- После сходимости определить принадлежность x_i к кластерам:

$$\text{clust}_i := \arg \max_{c \in C} g_{nc}.$$

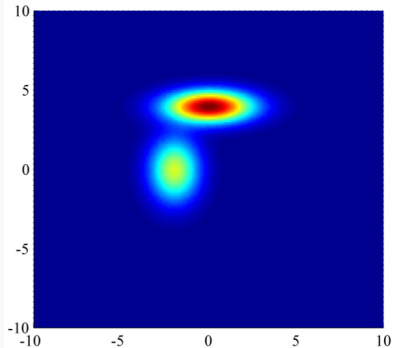
ПОЧЕМУ ТАК?

- Как доказать, что Е-шаг и М-шаг действительно в данном случае так выглядят?
- На Е-шаге для параметров $\theta = (\mathbf{w}, \mu, \sigma)$:

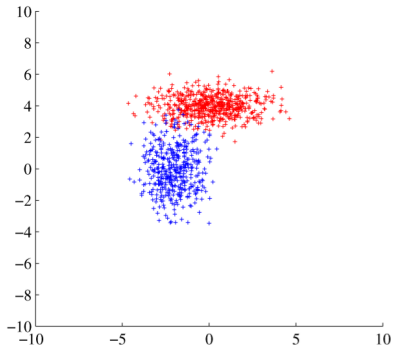
$$\begin{aligned} Q(\theta \mid \theta^{(m)}) &= \mathbb{E}_{Z|Y, \theta^{(m)}} [\log p(Y, Z \mid \theta)] = \\ &= \mathbb{E}_{\theta^{(m)}} \left[\log \prod_{n=1}^N \prod_{c=1}^C p(\mathbf{y}_n, z_{nc} \mid \theta)^{z_{nc}} \right] = \\ &= \mathbb{E}_{\theta^{(m)}} \left[\sum_{n=1}^N \sum_{c=1}^C z_{nc} (\log p(z_{nc} \mid w_c) + \log p(\mathbf{y}_n \mid \mu_c, \sigma_c)) \right] = \\ &= \sum_{n=1}^N \sum_{c=1}^C (\mathbb{E}_{\theta^{(m)}} [z_{nc}] \log w_c + \mathbb{E}_{\theta^{(m)}} [z_{nc}] \log p(\mathbf{y}_n \mid \mu_c, \sigma_c)). \end{aligned}$$

- Отсюда и получается обучение каждого гауссиана независимо, но с весами $g_{nc} = \mathbb{E}_{\theta^{(m)}} [z_{nc}]$.

True GMM density

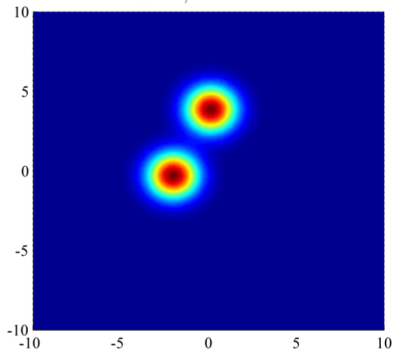


1000 i.i.d. samples



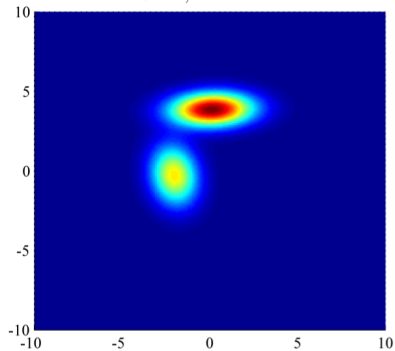
Initial Guess

$$m = 0, L^{(0)} = -3.9756$$



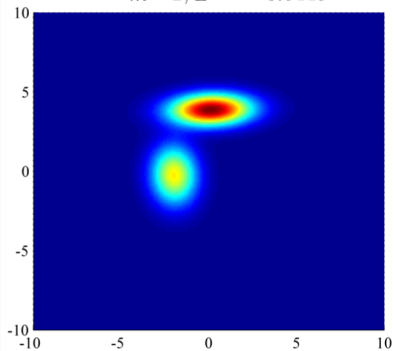
1st EM estimate

$$m = 1, L^{(1)} = -3.6492$$



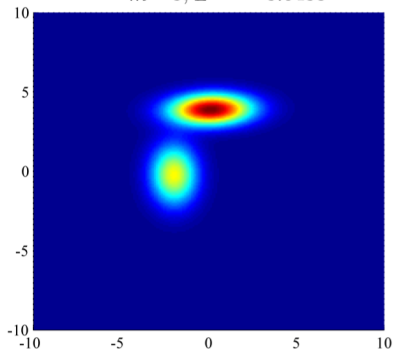
2nd EM estimate

$$m = 2, L^{(2)} = -3.6446$$



3rd EM estimate

$$m = 3, L^{(3)} = -3.6438$$



- Остается проблема: нужно задавать количество кластеров.
- Как её решать?

- Один из самых известных алгоритмов кластеризации – алгоритм k -средних – это фактически упрощение алгоритма EM.
- Формальная цель алгоритма k -средних — минимизировать меру ошибки

$$E(Y, C) = \sum_{n=1}^n \|\mathbf{y}_n - \mu_i\|^2,$$

где μ_i — ближайший к $\mathbf{y}_n$ центр кластера.

- Т.е. мы не относим точки к кластерам, а двигаем центры, а принадлежность точек определяется автоматически.

- Идея та же, что в EM:
 - Проинициализировать.
 - Классифицировать точки по ближайшему к ним центру кластера.
 - Перевычислить каждый из центров.
 - Если ничего не изменилось, остановиться, если изменилось — повторить.

kMeans($Y, |C|$):

- Инициализировать центры $|C|$ кластеров $\mu_1, \dots, \mu_{|C|}$.
- Пока принадлежность кластерам не перестанет изменяться:
 - Определить принадлежность $\mathbf{y}_n$ к кластерам:

$$\text{clust}_n := \arg \min_{c \in C} \rho(\mathbf{y}_n, \mu_c).$$

- Определить новое положение центров кластеров:

$$\mu_c := \frac{\sum_{\text{clust}_n=c} \mathbf{y}_n}{\sum_{\text{clust}_n=c} 1}.$$

И чем же это от EM отличается?

- Разница в том, что мы не считаем вероятности принадлежности кластерам, а жестко приписываем каждый объект одному кластеру.
- Это на самом деле вариант EM, в котором вместо полного распределения $p(X | \theta)$, которое в Q-функции используется, мы берём просто точку максимального правдоподобия
- Point-estimate variant of EM, или Classification EM:

$$\begin{aligned}\mathbf{z}^{(m)} &= \arg \max_{\mathbf{z}} p(\mathbf{z} | \mathbf{y}, \theta^{(m)}), \\ \theta^{(m+1)} &= \arg \max_{\theta} p(\mathbf{z}^{(m)} | \theta^{(m)}).\end{aligned}$$

ОБОСНОВАНИЕ АЛГОРИТМА ЕМ

- Дадим формальное обоснование алгоритма ЕМ.
- Мы решаем задачу максимизации правдоподобия по данным $\mathbf{y} = \{y_1, \dots, y_N\}$.

$$L(\theta | Y) = p(Y | \theta) = \prod p(y_i | \theta)$$

или, что то же самое, максимизации $\ell(\theta | Y) = \log L(\theta | Y)$.

- ЕМ может помочь, если этот максимум трудно найти аналитически.

- Давайте предположим, что в данных есть *скрытые компоненты*, такие, что если бы мы их знали, задача была бы проще.
- Замечание: совершенно не обязательно эти компоненты должны иметь какой-то физический смысл. :) Может быть, так просто удобнее.
- В любом случае, получается набор данных $X = (Y, Z)$ с совместной плотностью

$$p(\mathbf{x} \mid \theta) = p(\mathbf{y}, \mathbf{z} \mid \theta) = p(\mathbf{z} \mid \mathbf{y}, \theta)p(\mathbf{y} \mid \theta).$$

- Получается полное правдоподобие $L(\theta \mid X) = p(Y, Z \mid \theta)$. Это случайная величина (т.к. Z неизвестно).

- Заметим, что настоящее правдоподобие $L(\theta) = E_Z [p(Y, Z | \theta) | Y, \theta]$.
- Е-шаг алгоритма ЕМ вычисляет условное ожидание (логарифма) полного правдоподобия при условии Y и текущих оценок параметров θ_n :

$$Q(\theta, \theta_n) = E [\log p(Y, Z | \theta) | Y, \theta_n].$$

- Здесь θ_n — текущие оценки, а θ — неизвестные значения (которые мы хотим получить в конечном счёте); т.е. $Q(\theta, \theta_n)$ — это функция от θ .

- Е-шаг алгоритма ЕМ вычисляет условное ожидание (логарифма) полного правдоподобия при условии Y и текущих оценок параметров θ :

$$Q(\theta, \theta_n) = E [\log p(Y, Z | \theta) | Y, \theta_n].$$

- Условное ожидание — это

$$E [\log p(Y, Z | \theta) | Y, \theta_n] = \int_{\mathbf{z}} \log p(Y, \mathbf{z} | \theta) p(\mathbf{z} | Y, \theta_n) d\mathbf{z},$$

где $p(\mathbf{z} | Y, \theta_n)$ — маргинальное распределение скрытых компонентов данных.

- ЕМ лучше всего применять, когда это выражение легко подсчитать, может быть, даже аналитически.
- Вместо $p(\mathbf{z} | Y, \theta_n)$ можно подставить $p(\mathbf{z}, Y | \theta_n) = p(\mathbf{z} | Y, \theta_n)p(Y | \theta_n)$, от этого ничего не изменится.

- В итоге после Е-шага алгоритма ЕМ мы получаем функцию $Q(\theta, \theta_n)$.
- На М-шаге мы максимизируем

$$\theta_{n+1} = \arg \max_{\theta} Q(\theta, \theta_n).$$

- Затем повторяем процедуру до сходимости.
- В принципе, достаточно просто находить θ_{n+1} , для которого $Q(\theta_{n+1}, \theta_n) > Q(\theta_n, \theta_n)$ — Generalized EM.
- Осталось понять, что значит $Q(\theta, \theta_n)$ и почему всё это работает.

- Мы хотели перейти от θ_n к θ , для которого $\ell(\theta) > \ell(\theta_n)$.

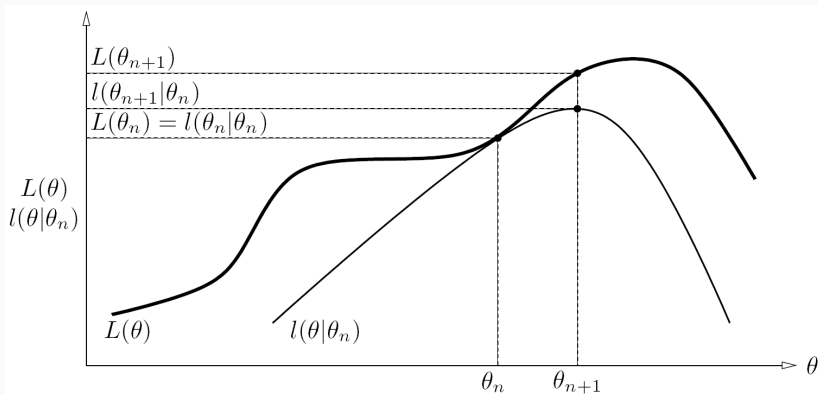
$$\begin{aligned}
 \ell(\theta) - \ell(\theta_n) &= \\
 &= \log \left(\int_{\mathbf{z}} p(Y | \mathbf{z}, \theta) p(\mathbf{z} | \theta) d\mathbf{z} \right) - \log p(Y | \theta_n) = \\
 &= \log \left(\int_{\mathbf{z}} p(\mathbf{z} | Y, \theta_n) \frac{p(Y | \mathbf{z}, \theta) p(\mathbf{z} | \theta)}{p(\mathbf{z} | Y, \theta_n)} d\mathbf{z} \right) - \log p(Y | \theta_n) \geq \\
 &\geq \int_{\mathbf{z}} p(\mathbf{z} | Y, \theta_n) \log \left(\frac{p(Y | \mathbf{z}, \theta) p(\mathbf{z} | \theta)}{p(\mathbf{z} | Y, \theta_n)} \right) d\mathbf{z} - \log p(Y | \theta_n) = \\
 &= \int_{\mathbf{z}} p(\mathbf{z} | Y, \theta_n) \log \left(\frac{p(Y | \mathbf{z}, \theta) p(\mathbf{z} | \theta)}{p(Y | \theta_n) p(\mathbf{z} | Y, \theta_n)} \right) d\mathbf{z}.
 \end{aligned}$$

- Получили

$$\begin{aligned}\ell(\theta) &\geq \mathcal{L}(\theta, \theta_n) = \\ &= \ell(\theta_n) + \int_{\mathbf{z}} p(\mathbf{z} \mid Y, \theta_n) \log \left(\frac{p(Y \mid \mathbf{z}, \theta)p(\mathbf{z} \mid \theta)}{p(Y \mid \theta_n)p(\mathbf{z} \mid Y, \theta_n)} \right) d\mathbf{z}.\end{aligned}$$

Упражнение. Докажите, что $\mathcal{L}(\theta_n, \theta_n) = \ell(\theta_n)$.

- Иначе говоря, мы нашли нижнюю оценку на $\ell(\theta)$ везде, касание происходит в точке θ_n .
- Т.е. мы нашли нижнюю оценку для правдоподобия и смещаемся в точку, где она максимальна (или хотя бы больше текущей).
- Такая общая схема называется *ММ-алгоритм* (minorization-maximization). Мы к ним ещё вернёмся.



- Осталось только понять, что максимизировать можно Q .

$$\begin{aligned}\theta_{n+1} &= \arg \max_{\theta} l(\theta, \theta_n) = \arg \max_{\theta} \left\{ \ell(\theta_n) + \right. \\ &\quad \left. + \int_{\mathbf{z}} f(y | X, \theta_n) \log \left(\frac{p(X | y, \theta) f(y | \theta)}{p(X | \theta_n) f(y | X, \theta_n)} \right) d\mathbf{z} \right\} = \\ &= \arg \max_{\theta} \left\{ \int_{\mathbf{z}} p(\mathbf{z} | X, \theta_n) \log (p(X | y, \theta) p(\mathbf{z} | \theta)) d\mathbf{z} \right\} = \\ &= \arg \max_{\theta} \left\{ \int_{\mathbf{z}} p(\mathbf{z} | X, \theta_n) \log p(X, y | \theta) d\mathbf{z} \right\} = \\ &= \arg \max_{\theta} \{Q(\theta, \theta_n)\},\end{aligned}$$

а остальное от θ не зависит. Вот и получился ЕМ.

ИСТОРИЯ И ПЕРВЫЕ ПРИМЕНЕНИЯ

- Всё это появилось в работе Dempster–Laird–Rubin; доклад на Royal Statistical Society в 1976, статья вышла в 1977



Maximum Likelihood from Incomplete Data via the *EM* Algorithm

By A. P. DEMPSTER, N. M. LAIRD and D. B. RUBIN

Harvard University and Educational Testing Service

[Read before the ROYAL STATISTICAL SOCIETY at a meeting organized by the RESEARCH SECTION on Wednesday, December 8th, 1976, Professor S. D. SILVEY in the Chair]

SUMMARY

A broadly applicable algorithm for computing maximum likelihood estimates from incomplete data is presented at various levels of generality. Theory showing the monotone behaviour of the likelihood and convergence of the algorithm is derived. Many examples are sketched, including missing value situations, applications to grouped, censored or truncated data, finite mixture models, variance component estimation, hyperparameter estimation, iteratively reweighted least squares and factor analysis.

Keywords: MAXIMUM LIKELIHOOD; INCOMPLETE DATA; EM ALGORITHM; POSTERIOR MODE

- Но были и более ранние аналоги (сами DLR тоже пишут, что было много примеров раньше, и ссылаются на них)...

- (Ceppellini et al., 1955): подсчёт частот генов в популяции:
 - в популяции есть k аллелей с частотами $p_1, \dots, p_k$ и генами $G_1, \dots, G_k$;
 - например, в рассмотренной MN-системе есть два аллеля $G_1 = M$ и $G_2 = N$, и у человека диплоидные клетки, т.е. бывают варианты MM, MN и NN; если бы мы умели различать все три варианта, то было бы легко подсчитать гены каждого типа;
 - но что если M доминантный, а N рецессивный, т.е. гомозиготные MM- и гетерозиготные MN-организмы неразличимы?
 - есть закон Харди-Вайнберга, который говорит, что гомозиготы и гетерозиготы встречаются с частотами $\frac{p^2}{p^2+2pq}$ и $\frac{2pq}{p^2+2pq}$, где p и $q = 1 - p$ — частоты генов;
 - можно было бы всё подсчитать, но получается замкнутый круг: нужно знать частоты генов, чтобы посчитать долю MM и MN, но чтобы посчитать частоты, нужно знать долю MM и MN...

- И тут Cerrellini et al. говорят:

by counting genes we can obtain estimates p' and q' of the gene frequencies. Unfortunately, this argument is still circular, since it presupposes a knowledge of the gene frequencies in order to obtain the estimate. But if any value is provisionally assumed for p , say $p(1)$, the new estimate $p' = p(2)$ obtained by gene counting will be rather more accurate, since the number of genes in the recessive individuals is known for certain, and the provisional value $p(1)$ is used only in estimating the number of genes among the dominants. This new value $p(2)$ can be taken as a new provisional value, and a further estimate $p'(2) = p(3)$ obtained by gene counting. The last process can be continued, giving a series of values $p(1), p(2), p(3), \dots$, each more accurate than the last; when two successive values are equal to the order of accuracy desired their common value can be taken as the final estimate.

- Это, видимо, одно из самых ранних применений ЕМ-алгоритма, и такие применения актуальны, конечно, до сих пор.

СВОЙСТВА И ПРОСТЫЕ РАСШИРЕНИЯ ЕМ

- Что требуется, чтобы EM-алгоритм работал?
- Неформально нужно, чтобы $p(X | \theta)$ было легко максимизировать.
- А формально нужно, чтобы $p(\mathbf{y} | \mathbf{x}, \theta) = p(\mathbf{y} | \mathbf{x})$, т.е. чтобы выполнялось марковское свойство для $\theta \rightarrow \mathbf{x} \rightarrow \mathbf{y}$.
- На самом деле обычно EM применяется тогда, когда $\mathbf{y} = f(\mathbf{x})$ для детерминированной функции f , и это свойство тривиально выполняется.
- Более того, обычно EM применяется, когда f — это просто проекция, т.е. когда $\mathbf{x} = (\mathbf{y}, \mathbf{z})$, как мы изначально и рассматривали.

- Важное простое свойство — если данные состоят из независимо порождённых $X = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ (как всегда и бывает), то

$$\begin{aligned} Q(\theta \mid \theta^{(m)}) &= \mathbb{E}_{X|Y, \theta^{(m)}} \left[\log \prod_{n=1}^N p(\mathbf{x}_n \mid \theta) \right] = \\ &= \mathbb{E}_{X|Y, \theta^{(m)}} \left[\sum_{n=1}^N \log p(\mathbf{x}_n \mid \theta) \right] = \sum_{n=1}^N \mathbb{E}_{\mathbf{x}_n | \mathbf{y}_n, \theta^{(m)}} [\log p(\mathbf{x}_n \mid \theta)], \end{aligned}$$

потому что $p(\mathbf{x}_n \mid Y, \theta) = p(\mathbf{x}_n \mid \mathbf{y}_n, \theta)$

- Упражнение: докажите это!

- Другие обобщения могут пригодиться, если всё-таки подсчитать $\mathbb{E}_{X|Y, \theta^{(m)}} [\log p(X | \theta)]$ или оптимизировать $p(X | \theta)$ нелегко.
- Обобщённый EM (Generalized EM, GEM): вместо $\arg \max_{\theta} Q(\theta, \theta^{(m)})$ нам достаточно просто выбирать такую $\theta^{(m+1)}$, чтобы

$$Q(\theta^{(m+1)}, \theta^{(m)}) > Q(\theta^{(m)}, \theta^{(m)}).$$

- Стохастический EM (Stochastic EM): если Q-функцию не получается посчитать в замкнутой форме, но и просто максимум брать не хочется, как в Classification EM, можно попробовать брать $\mathbf{x}$ случайным образом на E-шаге:

$$\mathbf{x}^{(m)} \sim p(\mathbf{x} \mid \mathbf{y}, \theta^{(m)}),$$

а потом использовать его на M-шаге как обычно:

$$\theta^{(m+1)} = \arg \max_{\theta} p(\mathbf{x}^{(m)} \mid \theta^{(m)}).$$

- Монте-Карло EM (Monte Carlo EM): саму Q-функцию тоже можно попытаться приблизить, если подсчитать сложно; можно использовать приближение ожидания в Q-функции через сэмплирование:

$$Q(\theta \mid \theta^{(m)}) \approx \frac{1}{R} \sum_{r=1}^R \log p(\mathbf{x}^{(m,r)} \mid \theta), \text{ где } \mathbf{x}^{(m,r)} \sim p(\mathbf{x} \mid \mathbf{y}, \theta^{(m)}).$$

- Впрочем, в таких случаях часто можно вообще забыть на EM и аппроксимировать напрямую апостериорное распределение.

- А можно и априорное распределение в ЕМ добавить, конечно же:

$$\theta_{MAP} = \arg \max_{\theta} \log p(\theta \mid Z) = \arg \max_{\theta} (\log p(Y \mid \theta) + \log p(\theta)) .$$

- При этом базовая схема особо не меняется:

$$\begin{aligned} Q(\theta \mid \theta^{(m)}) &= \mathbb{E}_{X|Y, \theta^{(m)}} [\log p(X \mid \theta)] , \\ \theta^{(m+1)} &= \arg \max_{\theta} \left(Q(\theta \mid \theta^{(m)}) + \log p(\theta) \right) . \end{aligned}$$

- Например, так можно избежать вырожденных случаев (кластер из одной точки).

- И EM, и k -means хорошо обобщаются на случай частично обученных кластеров.
- То есть про часть точек уже известно, какому кластеру они принадлежат.
- Как это учесть?

- И EM, и k -means хорошо обобщаются на случай частично обученных кластеров.
- То есть про часть точек уже известно, какому кластеру они принадлежат.
- Как это учесть?
- Чтобы учесть информацию о точке $\mathbf{x}_i$, достаточно для EM положить скрытую переменную g_{nc} равной тому кластеру, которому нужно, с вероятностью 1, а остальным — с вероятностью 0, и не пересчитывать.
- Для k -means то же самое, но для clust_i .

ПРИМЕР: МОДЕЛИ БРЭДЛИ--ТЕРРИ

- Другой подход к рейтинг-системам — модели Брэдли–Терри (Bradley–Terry).
- Модель предполагает, что для участников $1, \dots, n$ можно подобрать такие рейтинги γ_i , $i = 1..n$, что вероятность победы участника i над участником j равна

$$p(i \text{ побеждает } j) = \frac{\gamma_i}{\gamma_i + \gamma_j}.$$

- Основная задача заключается в том, чтобы найти $\gamma = (\gamma_1, \dots, \gamma_m)$ максимального правдоподобия из имеющихся данных D .

- Если принять априорное распределение равномерным, можно просто максимизировать правдоподобие

$$p(D|\gamma) = \prod_{i=1}^m \prod_{j=1}^m \left(\frac{\gamma_i}{\gamma_i + \gamma_j} \right)^{w_{ij}},$$

где w_{ij} — то, сколько раз x_i обыграл x_j при их попарном сравнении ($w_{ii} = 0$ по определению).

- Взяв логарифм, будем максимизировать

$$l(\gamma) = \sum_{i=1}^m \sum_{j=1}^m (w_{ij} \log \gamma_i - w_{ij} \log(\gamma_i + \gamma_j)).$$

- Максимизировать будем классическим ММ-алгоритмом (minorization-maximization), фактически вариационным приближением.
- Заметим, что

$$1 + \log \frac{x}{y} - \frac{x}{y} \leq 0.$$

Упражнение. Докажите это.

- Рассмотрим вспомогательную функцию

$$Q(\gamma, \gamma^{(k)}) = \sum_{i,j} w_{ij} \left[\log \gamma_i - \frac{\gamma_i + \gamma_j}{\gamma_i^{(k)} + \gamma_j^{(k)}} - \log (\gamma_i^{(k)} + \gamma_j^{(k)}) + 1 \right].$$

Упражнение. Используя предыдущее неравенство, докажите, что

$$Q(\gamma, \gamma^{(k)}) \leq l(\gamma).$$

- Чтобы найти $\max_{\gamma} Q(\gamma, \gamma^{(k)})$, можно просто взять производные $\frac{\partial Q}{\partial \gamma_l}$:

$$\begin{aligned}\frac{\partial Q}{\partial \gamma_l} &= \sum_{i,j} w_{ij} \left[\frac{\delta_{il}}{\gamma_i} - \frac{\delta_{il} + \delta_{jl}}{\gamma_i^{(k)} + \gamma_j^{(k)}} \right] = \\ &= \frac{1}{\gamma_l} \sum_j w_{lj} - \sum_j \frac{w_{lj}}{\gamma_i^{(k)} + \gamma_j^{(k)}} - \sum_i \frac{w_{il}}{\gamma_i^{(k)} + \gamma_j^{(k)}}.\end{aligned}$$

- Если w_l — общее количество побед игрока l ($w_l = \sum_j w_{lj}$), и N_{ij} — количество встреч между игроками i и j ($N_{ij} = w_{ij} + w_{ji}$), получаем

$$\frac{w_l}{\gamma_l} - \sum_j \frac{N_{lj}}{\gamma_l^{(k)} + \gamma_j^{(k)}} = 0.$$

- В результате правило пересчёта на одной итерации выглядит так:

$$\gamma_l^{(k+1)} := w_l \left[\sum_j \frac{N_{lj}}{\gamma_l^{(k)} + \gamma_j^{(k)}} \right]^{-1}.$$

- Получили алгоритм оценки рейтингов. Правда, он пока работает только для ситуации, когда игроки встречаются один на один и выигрывают или проигрывают.
- Но даже в шахматах бывают ничьи. Как их учесть?

- Если возможны ничьи, их вероятность можно описать дополнительным параметром $\theta > 1$, и это приводит к модели, в которой

$$p(i \text{ побеждает } j) = \frac{\gamma_i}{\gamma_i + \theta \gamma_j},$$

$$p(j \text{ побеждает } i) = \frac{\gamma_j}{\theta \gamma_i + \gamma_j},$$

$$p(i \text{ и } j \text{ играют вничью}) = \frac{(\theta^2 - 1)\gamma_i\gamma_j}{(\gamma_i + \theta\gamma_j)(\theta\gamma_i + \gamma_j)}.$$

- Аналогично можно вводить другие обобщения.
- Например, если результат может зависеть от порядка элементов в паре (скажем, команды проводят «домашние» матчи и «гостевые»), можно ввести дополнительный параметр θ , характеризующий, насколько большее преимущество дают «родные стены», и рассмотреть модель с

$$p(i \text{ побеждает } j) = \begin{cases} \frac{\theta\gamma_i}{\theta\gamma_i + \gamma_j}, & \text{если } i \text{ играет дома,} \\ \frac{\gamma_j}{\theta\gamma_i + \gamma_j}, & \text{если } j \text{ играет дома.} \end{cases}$$

- Можно даже обобщить на случай, когда в одном турнире встречаются несколько игроков: пусть перестановка π подмножества игроков $A = \{1, \dots, k\}$ (результат турнира) имеет вероятность

$$p_A(\pi) = \prod_{i=1}^k \frac{\gamma_{\pi(i)}}{\gamma_{\pi(i)} + \gamma_{\pi(i+1)} + \dots + \gamma_{\pi(k)}}.$$

- Можно показать, что такая модель эквивалентна весьма естественной «аксиоме Люса»: для любой модели, в которой вероятности игроков обыграть друг друга в любой паре не равны нулю, для любых подмножеств игроков $A \subset B$ и любого игрока $i \in A$

$$\begin{aligned} p_B(i \text{ побеждает}) &= \\ &= p_A(i \text{ побеждает}) p_B(\text{побеждает кто-то из множества } A). \end{aligned}$$

- Но для крупных турниров это перестаёт работать; и совсем трудно что-то осмысленное сделать, если игроки соревнуются не поодиночке, а в командах.

СПАСИБО!

Спасибо за внимание!

