КЛАСТЕРИЗАЦИЯ И EM-АЛГОРИТМ

Сергей Николенко

СПбГУ — Санкт-Петербург

25 февраля 2025 г.

Random facts:

- 25 февраля 1831 г. в рамках подавления польского восстания 1830 года состоялось сражение при Грохове между русскими и войсками повстанцев; потери русских составили 9400 человек, поляки лишились 12 000 человек и трёх орудий; польские войска проиграли битву, но взять Варшаву одним ударом не удалось, и фельдмаршал Дибич-Забайкальский был вынужден отступить к базам снабжения
- 25 февраля 1956 г. в ходе XX съезда КПСС Никита Хрущёв выступил с закрытым докладом «О культе личности и его последствиях»; один из очевидцев доклада А.Н. Яковлев вспоминал: «В зале стояла глубокая тишина. Не слышно было ни скрипа кресел, ни кашля, ни шёпота. Никто не смотрел друг на друга — то ли от неожиданности случившегося, то ли от смятения и страха. Шок был невообразимо глубоким»
- 25 февраля 1964 г. Мухаммед Али стал чемпионом мира в тяжёлом весе, нокаутировав Сонни Листона в седьмом раунде
- 25 февраля 1986 г. Фердинанд Маркос бежал из Филиппин на Гавайи после того, как смог победить на выборах Корасон Акино, вдову убитого конкурента, только с очевидными нарушениями; Акино стала президентом, а эти события стали известны как «жёлтая революция» — как ни странно, к «цветным революциям» её обычно не относят

Кластеризация

Суть лекции

- *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.

Чуть более формально

- 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.

Идея

- 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.

Алгоритм

`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.
  - $C := C \cup \{c_1c_2\} \quad \{c_1, c_2\}$.
- Выдаём G.

Результат

- 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?

Результат

- 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 vs. complete-link

- *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?

Очевидный алгоритм

- 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

- 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).

Кластеризация

- Как использовать минимальное остовное дерево для кластеризации?

Кластеризация

- 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.

DBSCAN

- 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.

DBSCAN

- Алгоритм DBSCAN:

Алгоритм 1. DBSCAN

```
foreach точка x ∈ D do
    if x не посещена then
        пометить x как посещённую;
        A ← { y ∈ D | ‖y − x‖ ≤ ε };
        if |A| < min_samples then
            пометить x как выброс/шум;
        else
            создать новый кластер C и добавить туда x;
            while A ≠ ∅ do
                выбрать точку y ∈ A, удалить y из A;
                if y не посещена then
                    пометить y как посещённую;
                    A′ ← { y′ ∈ D | ‖y′ − y‖ ≤ ε };
                    if |A′| ≥ min_samples then
                        A ← A ∪ A′;
                if y не принадлежит ни одному кластеру then
                    добавить y в кластер C;
```

DBSCAN

- Вариант DBSCAN — OPTICS (Ordering Points To Identify the Clustering Structure); для данных, где кластеры могут иметь разную плотность:
  - сначала проходим по всем точкам и вычисляем окрестность по заданному радиусу ϵ (это теперь максимальный порог);
  - если у точки достаточно соседей (больше $\mathrm{min\_samples}$), то она считается основной; но теперь для каждой точки ещё рассчитываются
    - *основное расстояние* (core distance), расстояние от точки до её $\mathrm{min\_samples}$-го ближайшего соседа;
    - *расстояние достижимости* (reachability distance): минимальное расстояние, с которым точка может быть достигнута из основной точки; это значение обновляется во время обхода и показывает, насколько легко можно присоединить точку к уже существующему кластеру
- В результате у OPTICS получается упорядоченный список всех точек вместе с их значениями расстояния достижимости, и теперь можно кластеры выделять как угодно, а то и дерево строить

Алгоритм OPTICS

Алгоритм 2. OPTICS

Инициализировать для каждой $\boldsymbol{x} \in D$: Reachability($\boldsymbol{x}$) $\leftarrow$ undefined;
foreach *точка* $\boldsymbol{x} \in D$ **do**
  if $\boldsymbol{x}$ *не обработана* **then**
    пометить $\boldsymbol{x}$ как обработанную, добавить в список;
    $A \leftarrow \{\, \boldsymbol{y} \in D \mid \|\boldsymbol{y} - \boldsymbol{x}\| \le \epsilon \,\}$;
    вычислить core_dist($\boldsymbol{x}$) по множеству A;
    if core_dist($\boldsymbol{x}$) *определён* **then**
      инициализировать пустую приоритетную очередь Seeds;
      Update (Seeds, $A, \boldsymbol{x}$);
      while Seeds $\neq \varnothing$ **do**
        извлечь точку $\boldsymbol{y}$ с минимальным Reachability($\boldsymbol{y}$) из Seeds;
        пометить $\boldsymbol{y}$ как обработанную;
        $A' \leftarrow \{\, \boldsymbol{y}' \in D \mid \|\boldsymbol{y}' - \boldsymbol{y}\| \le \epsilon \,\}$;
        вычислить core_dist($\boldsymbol{y}$) по множеству A';
        добавить $\boldsymbol{y}$ в упорядоченный список точек;
        if core_dist($\boldsymbol{y}$) *определён* **then**
          Update (Seeds, $A', \boldsymbol{y}$);
Function `Update`(Seeds, A, $\boldsymbol{x}$)**:**
  $c =$ core_dist($\boldsymbol{x}$);
  foreach *точка* $\boldsymbol{y} \in A$ **do**
    if $\boldsymbol{y}$ *ещё не обработана* **then**
      $c' = \max(c, d(\boldsymbol{x}, \boldsymbol{y}))$;
      if Reachability($\boldsymbol{y}$) $=$ *undefined* **then**
        Reachability($\boldsymbol{y}$) $\leftarrow c'$;
        Seeds.insert($\boldsymbol{y}, c'$)
      else
        if c' < Reachability($\boldsymbol{y}$) **then**
          Reachability($\boldsymbol{y}$) $\leftarrow c'$;
          Seeds.move_up($\boldsymbol{y}, c'$)

Birch

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

Birch

- 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

Постановка задачи

- 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.

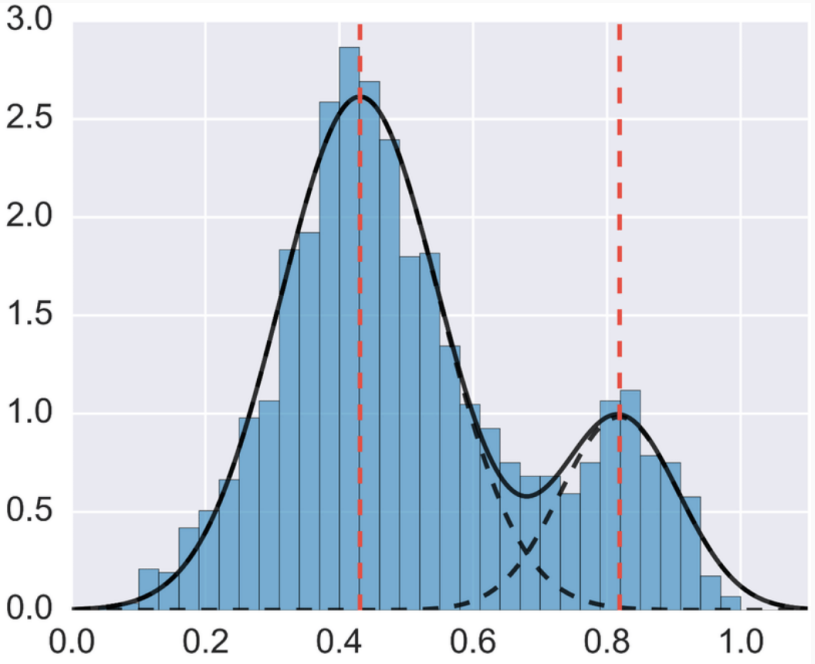
Постановка задачи

- 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 EM. 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?

Два распределения

- 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.

Суть алгоритма EM

- 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

- В примере с гауссианами:

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

- 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. :)

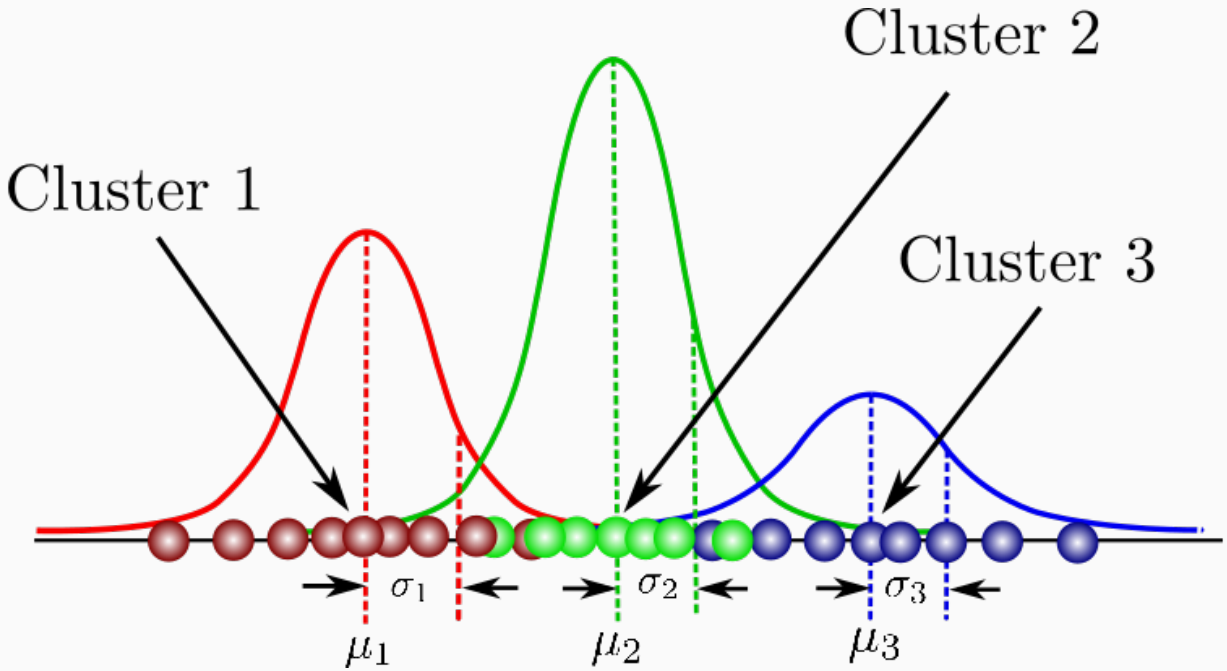
EM для кластеризации

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:



Гипотезы

- 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.

Гипотезы cont'd

- Остается вопрос: какими предположить распределения p_c?

Гипотезы cont'd

- 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.

Гипотезы cont'd

- 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 $\Sigma_c = \mathrm{diag}(\sigma_{c1}^2, \dots, \sigma_{c2}^2)$ (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}.

Идея алгоритма

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

Идея алгоритма

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

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

Идея алгоритма

- 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:

Идея алгоритма

- 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:

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

Идея алгоритма

- 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:

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

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

Алгоритм

`EMCluster`$(Y, |C|)$:

- Инициализировать $|C|$ кластеров; начальное приближение: $w_c := 1/|C|$, $\mu_c :=$ случайный $\mathbf{y}_n$, $\sigma^2_{cj} := \frac{1}{n|C|}\sum_{i=1}^{N}\left(y_{nj} - \mu_{cj}\right)^2$.
- 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: $w_c = \frac{1}{N}\sum_{i=1}^{N} g_{nc}$, $\mu_{cj} = \frac{1}{nw_c}\sum_{i=1}^{N} g_{nc} y_{nj}$,

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

- После сходимости определить принадлежность x_i к кластерам:

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

Почему так?

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

$$Q(\theta \mid \theta^{(m)}) = \mathbb{E}_{Z|Y,\theta^{(m)}} \left[\log p(Y, Z \mid \theta)\right] =$$

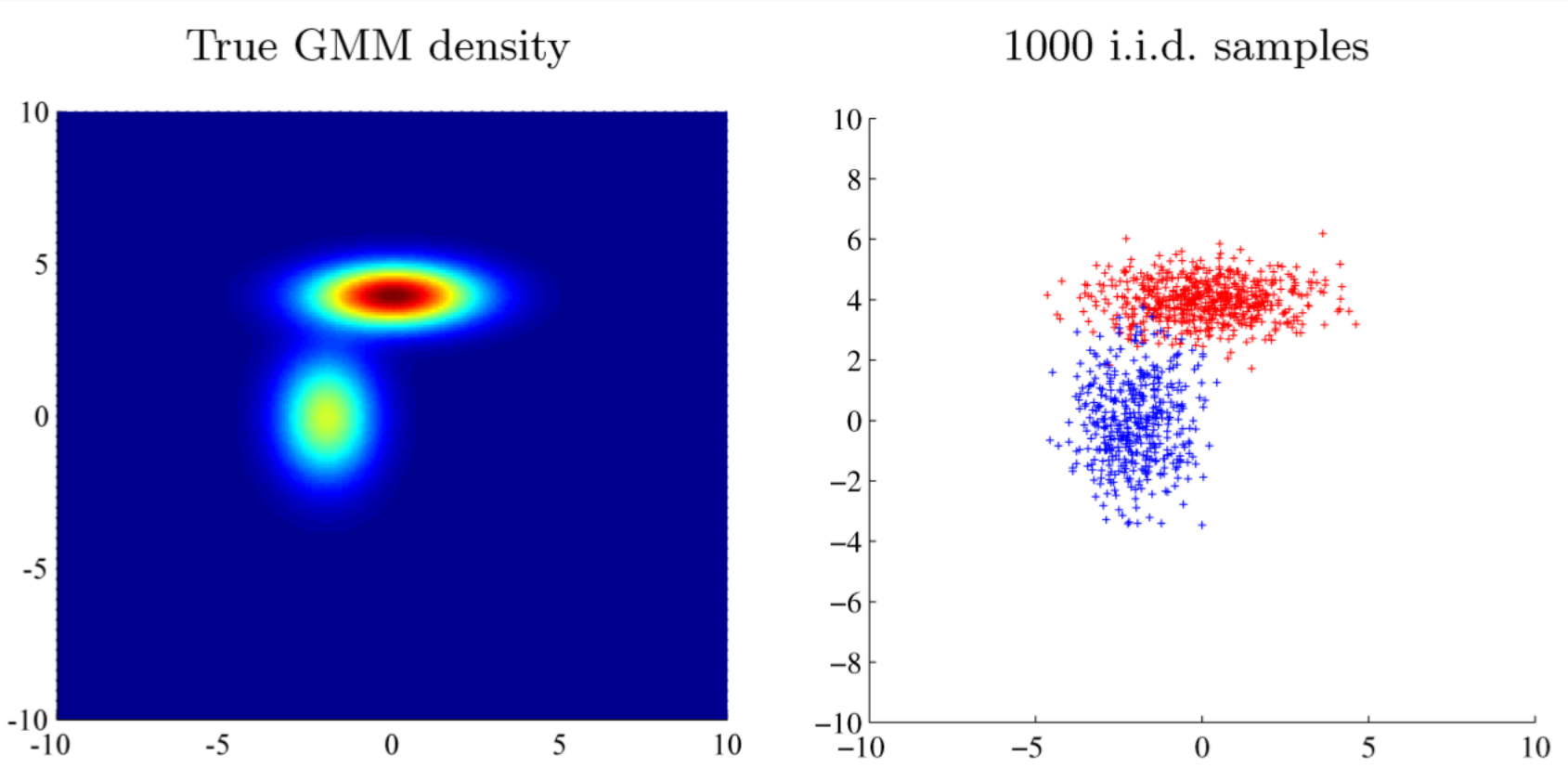
$$= \mathbb{E}_{\theta^{(m)}} \left[\log \prod_{n=1}^{N} \prod_{c=1}^{C} p(\mathbf{y}_n, z_{nc} \mid \theta)^{z_{nc}}\right] =$$

$$= \mathbb{E}_{\theta^{(m)}} \left[\sum_{n=1}^{N} \sum_{c=1}^{C} z_{nc} \left(\log p(z_{nc} \mid w_c) + \log p(\mathbf{y}_n \mid \mu_c, \sigma_c)\right)\right] =$$

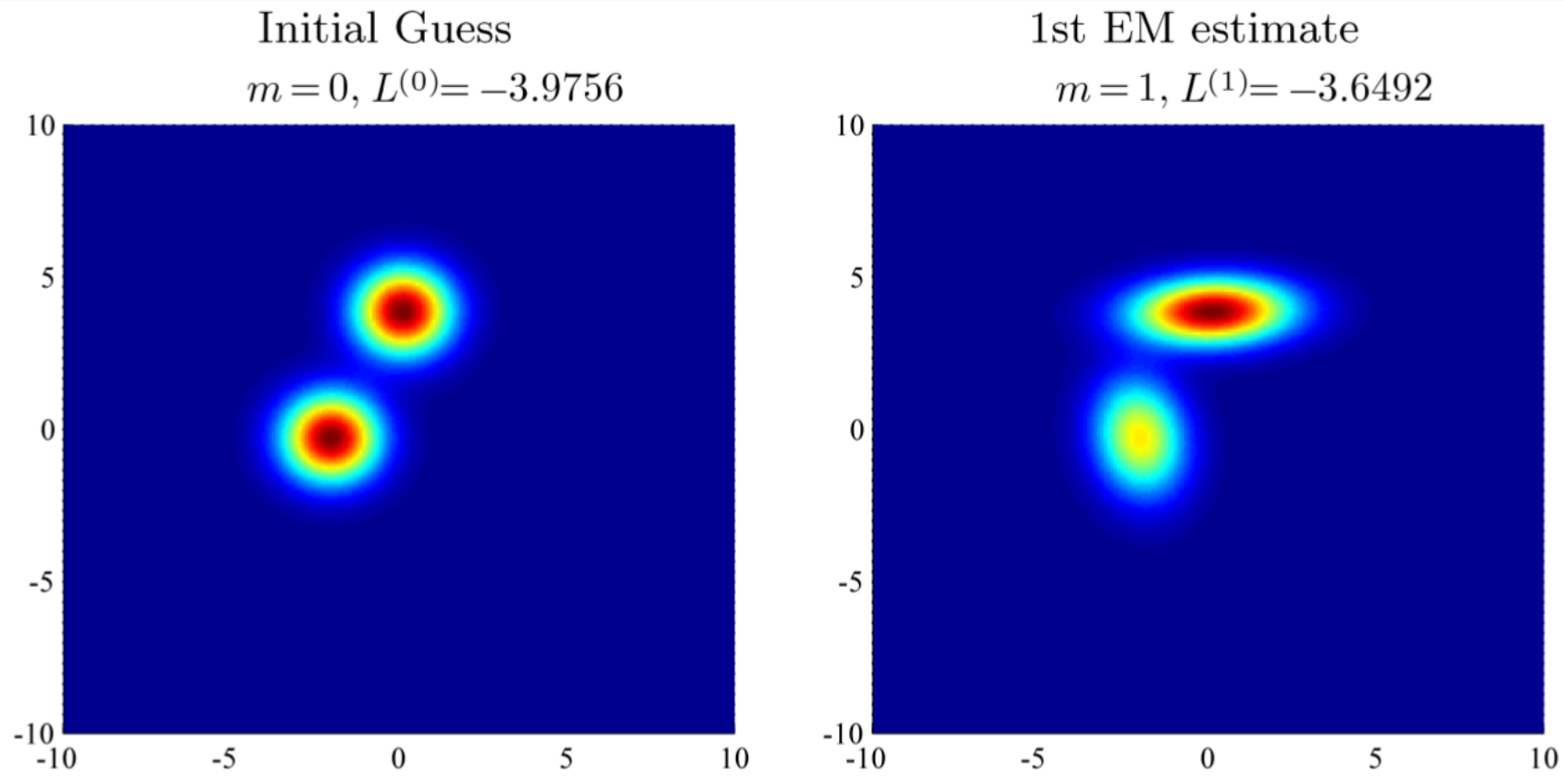
$$= \sum_{n=1}^{N} \sum_{c=1}^{C} \left(\mathbb{E}_{\theta^{(m)}} \left[z_{nc}\right] \log w_c + \mathbb{E}_{\theta^{(m)}} \left[z_{nc}\right] \log p(\mathbf{y}_n \mid \mu_c, \sigma_c)\right).$$

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

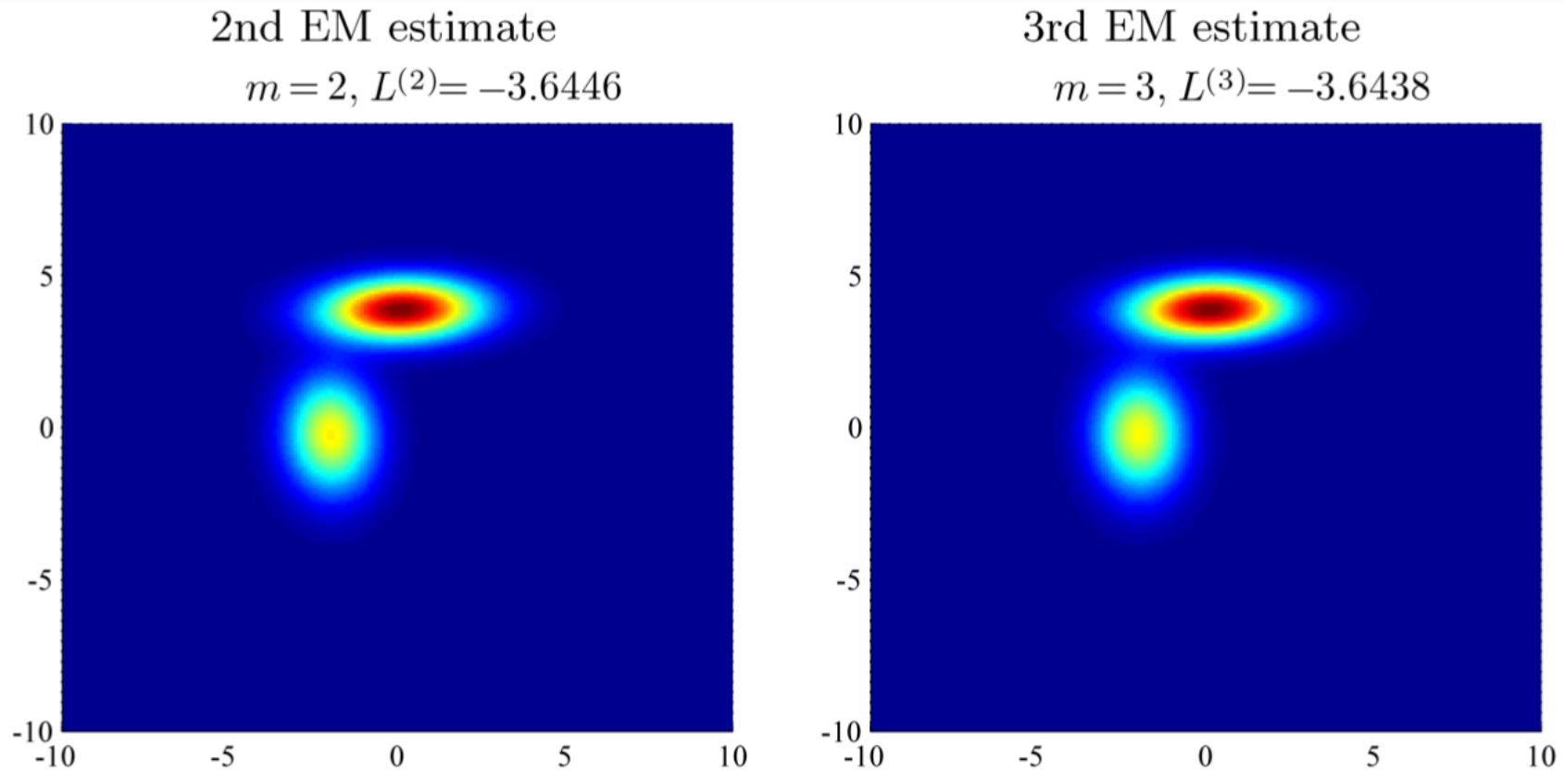
Пример


True GMM density
10
5
0
-5
-10
-10
-5
0
5
10
1000 i.i.d. samples
10
8
6
4
2
0
−2
−4
−6
−8
−10
−10
−5
0
5
10

Пример


Initial Guess
$m = 0$, $L^{(0)} = -3.9756$
1st EM estimate
$m = 1$, $L^{(1)} = -3.6492$
10
5
0
-5
-10
-10
-5
0
5
10

Пример


2nd EM estimate
$m = 2,\ L^{(2)} = -3.6446$
3rd EM estimate
$m = 3,\ L^{(3)} = -3.6438$
10
5
0
-5
-10
-10
-5
0
5
10

Проблема

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

Суть алгоритма k--средних

- 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.

Алгоритм неформально

- 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.

Алгоритм

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?

Point-estimate variant of EM

- 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 $p(X \mid \theta)$, которое в Q-функции используется, мы берём просто точку максимального правдоподобия
- Point-estimate variant of EM, или Classification EM:

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

Обоснование алгоритма EM

Обоснование алгоритма EM

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

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

  или, что то же самое, максимизации $\ell(\theta \mid Y) = \log L(\theta \mid Y)$.
- EM может помочь, если этот максимум трудно найти аналитически.

Обоснование алгоритма EM

- 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).

Обоснование алгоритма EM

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

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

- 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.

Обоснование алгоритма EM

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

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

- Условное ожидание — это

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

  где $p(\mathbf{z} \mid Y, \theta_n)$ — маргинальное распределение скрытых компонентов данных.
- EM лучше всего применять, когда это выражение легко подсчитать, может быть, даже аналитически.
- Вместо $p(\mathbf{z} \mid Y, \theta_n)$ можно подставить $p(\mathbf{z}, Y \mid \theta_n) = p(\mathbf{z} \mid Y, \theta_n) p(Y \mid \theta_n)$, от этого ничего не изменится.

Обоснование алгоритма EM

- В итоге после E-шага алгоритма EM мы получаем функцию $Q(\theta, \theta_n)$.
- На M-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.

Обоснование алгоритма EM

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

$$\ell(\theta) - \ell(\theta_n) =$$

$$= \log\left(\int_{\mathbf{z}} p(Y \mid \mathbf{z}, \theta) p(\mathbf{z} \mid \theta) \mathrm{d}\mathbf{z}\right) - \log p(Y \mid \theta_n) =$$

$$= \log\left(\int_{\mathbf{z}} p(\mathbf{z} \mid Y, \theta_n) \frac{p(Y \mid \mathbf{z}, \theta) p(\mathbf{z} \mid \theta)}{p(\mathbf{z} \mid Y, \theta_n)} \mathrm{d}\mathbf{z}\right) - \log p(Y \mid \theta_n) \geq$$

$$\geq \int_{\mathbf{z}} p(\mathbf{z} \mid Y, \theta_n) \log\left(\frac{p(Y \mid \mathbf{z}, \theta) p(\mathbf{z} \mid \theta)}{p(\mathbf{z} \mid Y, \theta_n)}\right) \mathrm{d}\mathbf{z} - \log p(Y \mid \theta_n) =$$

$$= \int_{\mathbf{z}} p(\mathbf{z} \mid Y, \theta_n) \log\left(\frac{p(Y \mid \mathbf{z}, \theta) p(\mathbf{z} \mid \theta)}{p(Y \mid \theta_n) p(\mathbf{z} \mid Y, \theta_n)}\right) \mathrm{d}\mathbf{z}.$$

Обоснование алгоритма EM

- Получили

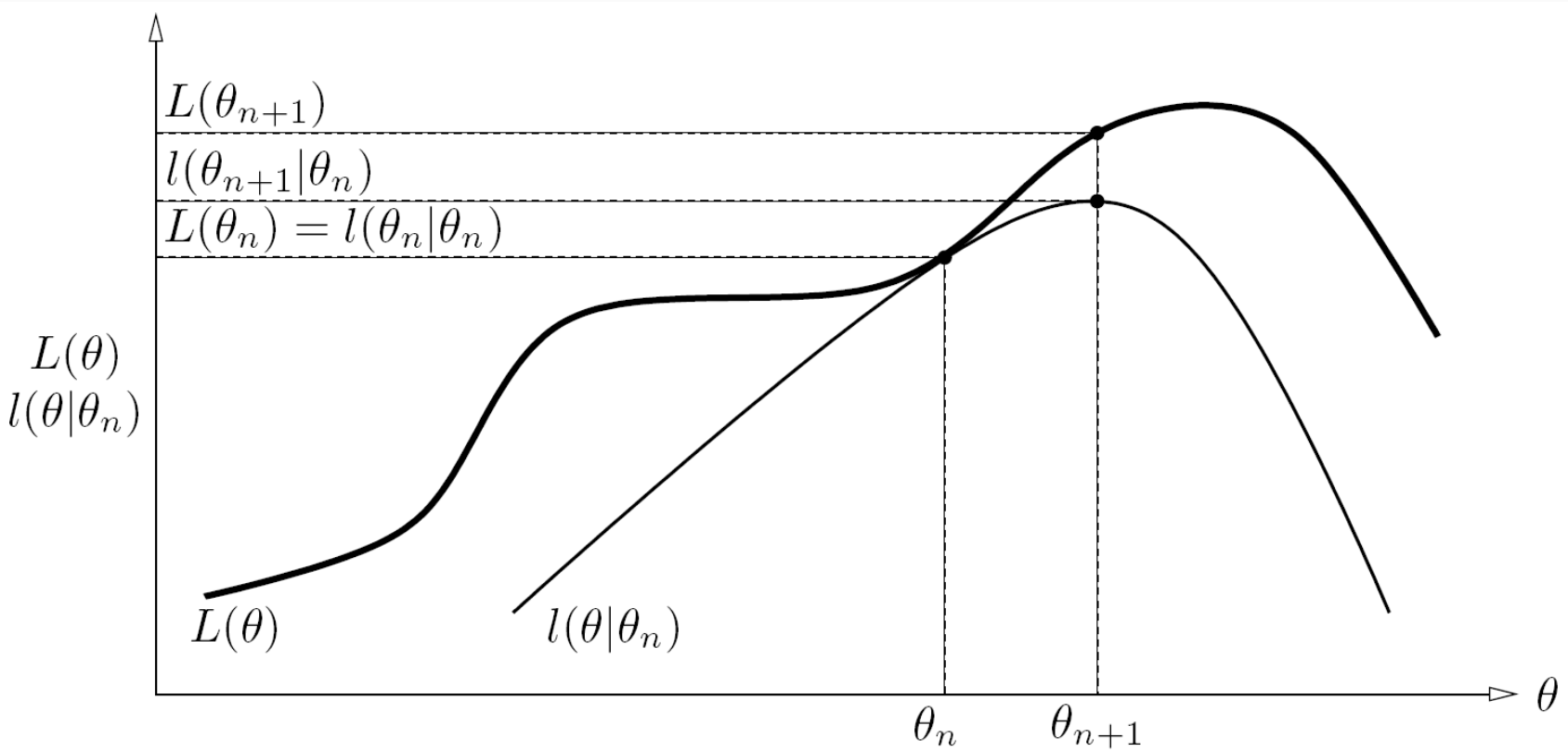
$$\ell(\theta) \geq \mathcal{L}(\theta, \theta_n) =$$
$$= \ell(\theta_n) + \int_{\mathbf{z}} p(\mathbf{z} \mid Y, \theta_n) \log \left(\frac{p(Y \mid \mathbf{z}, \theta) p(\mathbf{z} \mid \theta)}{p(Y \mid \theta_n) p(\mathbf{z} \mid Y, \theta_n)} \right) \mathrm{d}\mathbf{z}.$$

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

Обоснование алгоритма EM

- 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 *MM-алгоритм* (minorization-maximization). Мы к ним ещё вернёмся.

Обоснование алгоритма EM



Обоснование алгоритма EM

- Осталось только понять, что максимизировать можно Q.

$$\theta_{n+1} = \arg\max_\theta l(\theta, \theta_n) = \arg\max_\theta \{\ell(\theta_n) + \\ + \int_{\mathbf{z}} f(y \mid X, \theta_n) \log \left(\frac{p(X \mid y, \theta) f(y \mid \theta)}{p(X \mid \theta_n) f(y \mid X, \theta_n)} \right) \mathrm{d}\mathbf{z} \Bigg\} = \\ = \arg\max_\theta \left\{ \int_{\mathbf{z}} p(\mathbf{z} \mid X, \theta_n) \log \left(p(X \mid y, \theta) p(\mathbf{z} \mid \theta) \right) \mathrm{d}\mathbf{z} \right\} = \\ = \arg\max_\theta \left\{ \int_{\mathbf{z}} p(\mathbf{z} \mid X, \theta_n) \log p(X, y \mid \theta) \mathrm{d}\mathbf{z} \right\} = \\ = \arg\max_\theta \{Q(\theta, \theta_n)\},$$

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

История и первые применения

История EM-алгоритма

- 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)...

История EM-алгоритма

- (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...

История EM-алгоритма

- И тут Ceppellini 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 $p(1)$, $p(2)$, $p(3)$, ..., 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 EM-алгоритма, и такие применения актуальны, конечно, до сих пор.

Спасибо за внимание!