Кластеризация и EM-алгоритм

Сергей Николенко

НИУ ВШЭ — Санкт-Петербург

28 марта 2020 г.

Random facts:

- 28 марта — День освобождения тибетцев от крепостного рабства; 28 марта 1959 г. власти КНР объявили о роспуске тибетского правительства и провели военную операцию, в результате которой Далай-лама XIV и десятки тысяч других тибетцев бежали в Индию
- 28 марта 193 г. закончилось трёхмесячное правление императора Пертинакса; он недоплатил преторианцам, те его убили и выставили должность императора на аукцион; сенатор Дидий Юлиан победил в аукционе и правил ещё месяца два
- 28 марта 519 г. закончилась акакианская схизма — в 482 г. у константинопольского патриарха Акакия возникли разногласия с александрийским патриархом Иоанном, в 484 его низложили на соборе в Риме, но через 35 лет удалось (пока) помириться
- 28 марта 1939 г. войска генерала Франко заняли Мадрид после трёхлетней осады
- 28 марта 1979 г. произошла авария на станции Three Mile Island в Пенсильвании; 50% активной зоны реактора расплавились, но люди и окружающая среда не пострадали; тем не менее, на этом атомная энергетика в США фактически закончилась: с 1979 до 2012 года не было выдано ни одной новой лицензии на строительство АЭС

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 $X = \{x_1, \ldots, x_n\}$ 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 $x_1, x_2, \ldots, x_n$ в пространстве. Нужно кластеризовать.
- Считаем каждую точку кластером. Затем ближайшие точки объединяем, далее считаем единым кластером. Затем повторяем.
- Получается дерево.

Алгоритм

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

- 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.

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.

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 и кластеризация

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 x сэмплируется из суммы двух нормальных распределений. Дисперсии даны (одинаковые), нужно найти только средние μ_1, μ_2.

Два распределения

- Теперь нельзя понять, какие x_i были порождены каким распределением — классический пример *скрытых переменных*.
- Один тестовый пример полностью описывается как тройка $\langle x_i, z_{i1}, z_{i2} \rangle$, где $z_{ij} = 1$ iff x_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:

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

Мы подсчитываем эти ожидания, а потом подправляем гипотезу:

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

Обоснование алгоритма EM

- Дадим формальное обоснование алгоритма EM.
- Мы решаем задачу максимизации правдоподобия по данным $\mathcal{X} = \{x_1, \ldots, x_N\}$.

$$L(\theta \mid \mathcal{X}) = p(\mathcal{X} \mid \theta) = \prod p(x_i \mid \theta)$$

  или, что то же самое, максимизации $\ell(\theta \mid \mathcal{X}) = \log L(\theta \mid \mathcal{X})$.
- 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 $\mathcal{Z} = (\mathcal{X}, \mathcal{Y})$ с совместной плотностью

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

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

Обоснование алгоритма EM

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

$$Q(\theta, \theta_n) = E\left[\log p(\mathcal{X}, \mathcal{Y} \mid \theta) \mid \mathcal{X}, \theta_n\right].$$

- 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 $\mathcal{X}$ и текущих оценок параметров θ:

$$Q(\theta, \theta_n) = E\left[\log p(\mathcal{X}, \mathcal{Y} \mid \theta) \mid \mathcal{X}, \theta_n\right].$$

- Условное ожидание – это

$$E\left[\log p(\mathcal{X}, \mathcal{Y} \mid \theta) \mid \mathcal{X}, \theta_n\right] = \int_y \log p(\mathcal{X}, y \mid \theta) p(y \mid \mathcal{X}, \theta_n) \mathrm{d}y,$$

  где $p(y \mid \mathcal{X}, \theta_n)$ – маргинальное распределение скрытых компонентов данных.
- EM лучше всего применять, когда это выражение легко подсчитать, может быть, даже аналитически.
- Вместо $p(y \mid \mathcal{X}, \theta_n)$ можно подставить $p(y, \mathcal{X} \mid \theta_n) = p(y \mid \mathcal{X}, \theta_n) p(\mathcal{X} \mid \theta_n)$, 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 $\ell(\theta) > \ell(\theta_n)$.

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

$$= \log\left(\int_y p(\mathcal{X} \mid y, \theta) p(y \mid \theta) \mathrm{d}y\right) - \log p(\mathcal{X} \mid \theta_n) =$$

$$= \log\left(\int_y p(y \mid \mathcal{X}, \theta_n) \frac{p(\mathcal{X} \mid y, \theta) p(y \mid \theta)}{p(y \mid \mathcal{X}, \theta_n)} \mathrm{d}y\right) - \log p(\mathcal{X} \mid \theta_n) \geq$$

$$\geq \int_y p(y \mid \mathcal{X}, \theta_n) \log\left(\frac{p(\mathcal{X} \mid y, \theta) p(y \mid \theta)}{p(y \mid \mathcal{X}, \theta_n)}\right) \mathrm{d}y - \log p(\mathcal{X} \mid \theta_n) =$$

$$= \int_y p(y \mid \mathcal{X}, \theta_n) \log\left(\frac{p(\mathcal{X} \mid y, \theta) p(y \mid \theta)}{p(\mathcal{X} \mid \theta_n) p(y \mid \mathcal{X}, \theta_n)}\right) \mathrm{d}y.$$

Обоснование алгоритма EM

- Получили

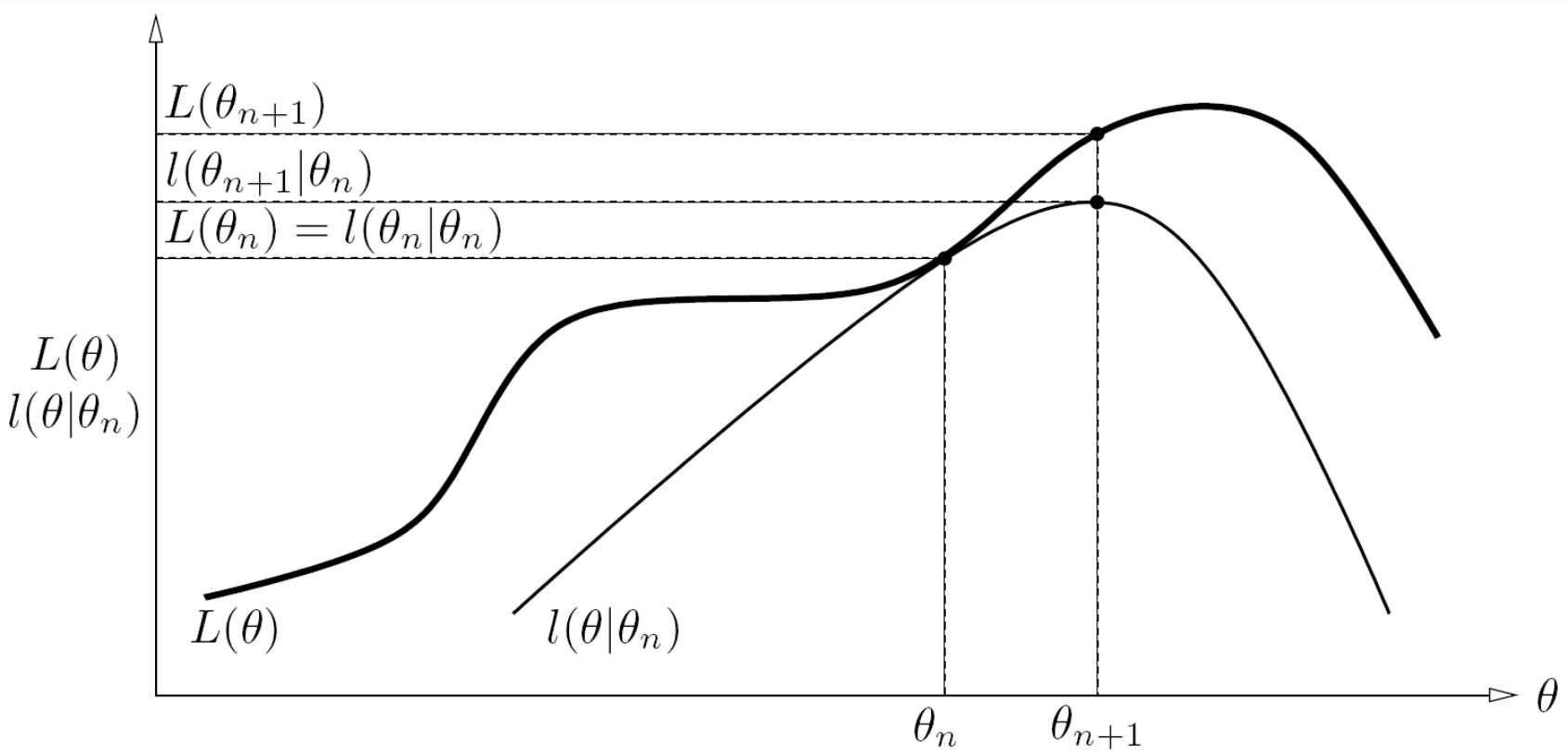
$$\ell(\theta) \geq l(\theta, \theta_n) =$$
$$= \ell(\theta_n) + \int_y p(y \mid \mathcal{X}, \theta_n) \log \left(\frac{p(\mathcal{X} \mid y, \theta) p(y \mid \theta)}{p(\mathcal{X} \mid \theta_n) p(y \mid \mathcal{X}, \theta_n)} \right) \mathrm{d}y.$$

Упражнение. Докажите, что $l(\theta_n, \theta_n) = \ell(\theta_n)$.

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, возможно, ещё вернёмся.

Обоснование алгоритма EM



Обоснование алгоритма EM

- Осталось только понять, что максимизировать можно Q.

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

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

Мысли?

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

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

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

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.
- Мы будем брать эллиптические гауссианы.
- *Гипотеза 2*: Каждый кластер c описывается d–мерной гауссовской плотностью с центром $\mu_c = \{\mu_{c1}, \ldots, \mu_{cd}\}$ и диагональной матрицей ковариаций $\Sigma_c = \mathrm{diag}(\sigma_{c1}^2, \ldots, \sigma_{c2}^2)$ (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 $(f_1(x), \ldots, f_n(x))$.
- Скрытые переменные в данном случае — вероятности g_{ic} того, что объект x_i принадлежит кластеру $c \in C$.

Идея алгоритма

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

Идея алгоритма

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

$$g_{ic} = \frac{w_c p_c(x_i)}{\sum_{c' \in C} w_{c'} p_{c'}(x_i)}.$$

Идея алгоритма

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

$$g_{ic} = \frac{w_c p_c(x_i)}{\sum_{c' \in C} w_{c'} p_{c'}(x_i)}.$$

- M–шаг: с использованием g_{ic} уточняются параметры кластеров w, μ, σ:

Идея алгоритма

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

$$g_{ic} = \frac{w_c p_c(x_i)}{\sum_{c' \in C} w_{c'} p_{c'}(x_i)}.$$

- M–шаг: с использованием g_{ic} уточняются параметры кластеров w, μ, σ:

$$w_c = \frac{1}{n} \sum_{i=1}^{n} g_{ic},$$

Идея алгоритма

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

$$g_{ic} = \frac{w_c p_c(x_i)}{\sum_{c' \in C} w_{c'} p_{c'}(x_i)}.$$

- *M*–шаг: с использованием g_{ic} уточняются параметры кластеров w, μ, σ:

$$w_c = \frac{1}{n} \sum_{i=1}^{n} g_{ic}, \quad \mu_{cj} = \frac{1}{n w_c} \sum_{i=1}^{n} g_{ic} f_j(x_i),$$

Идея алгоритма

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

$$g_{ic} = \frac{w_c p_c(x_i)}{\sum_{c' \in C} w_{c'} p_{c'}(x_i)}.$$

- M–шаг: с использованием g_{ic} уточняются параметры кластеров w, μ, σ:

$$w_c = \frac{1}{n} \sum_{i=1}^{n} g_{ic}, \quad \mu_{cj} = \frac{1}{n w_c} \sum_{i=1}^{n} g_{ic} f_j(x_i),$$

$$\sigma_{cj}^2 = \frac{1}{n w_c} \sum_{i=1}^{n} g_{ic} \left(f_j(x_i) - \mu_{cj}\right)^2.$$

Алгоритм

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

- Инициализировать $|C|$ кластеров; начальное приближение: $w_c := 1/|C|$, $\mu_c :=$ случайный x_i, $\sigma_{cj}^2 := \frac{1}{n|C|}\sum_{i=1}^{n}\left(f_j(x_i) - \mu_{cj}\right)^2$.
- Пока принадлежность кластерам не перестанет изменяться:
  - E–шаг: $g_{ic} := \frac{w_c p_c(x_i)}{\sum_{c' \in C} w_{c'} p_{c'}(x_i)}$.
  - M–шаг: $w_c = \frac{1}{n}\sum_{i=1}^{n} g_{ic}$, $\mu_{cj} = \frac{1}{nw_c}\sum_{i=1}^{n} g_{ic} f_j(x_i)$,

$$\sigma_{cj}^2 = \frac{1}{nw_c}\sum_{i=1}^{n} g_{ic}\left(f_j(x_i) - \mu_{cj}\right)^2.$$

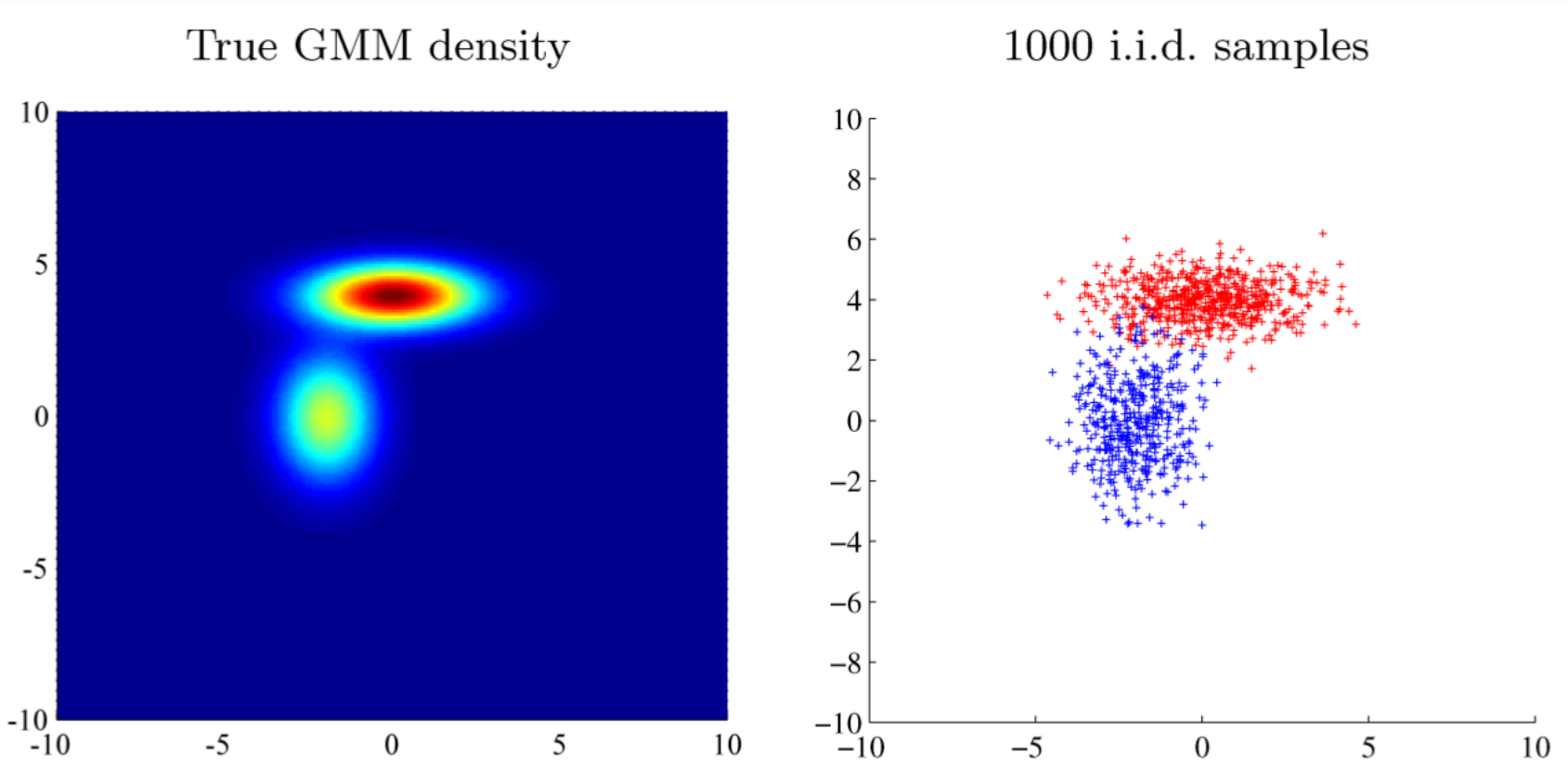
  - Определить принадлежность x_i к кластерам:

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

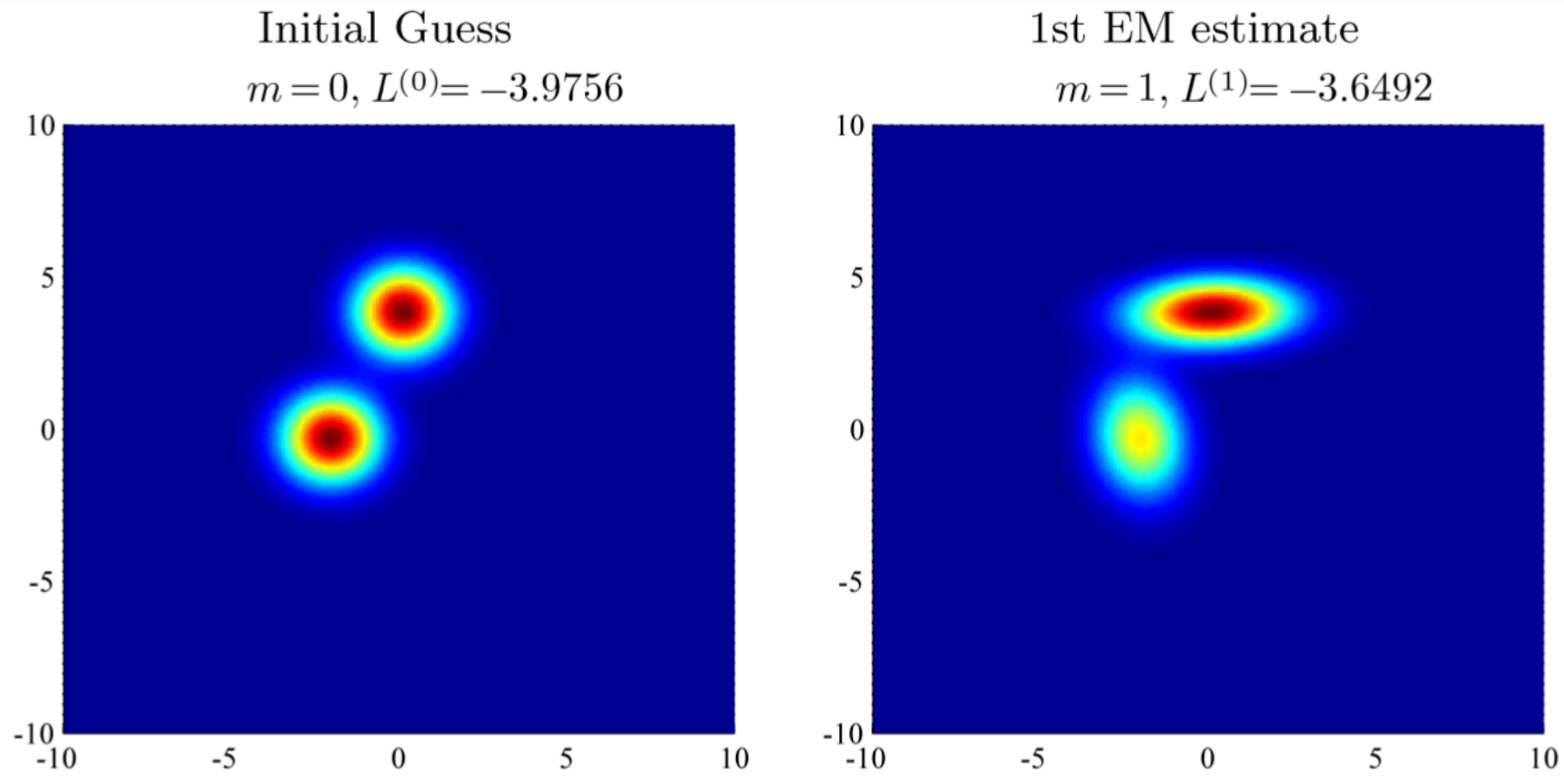
Упражнение

Упражнение. Докажите, что E-шаг и M-шаг действительно в данном случае так выглядят.

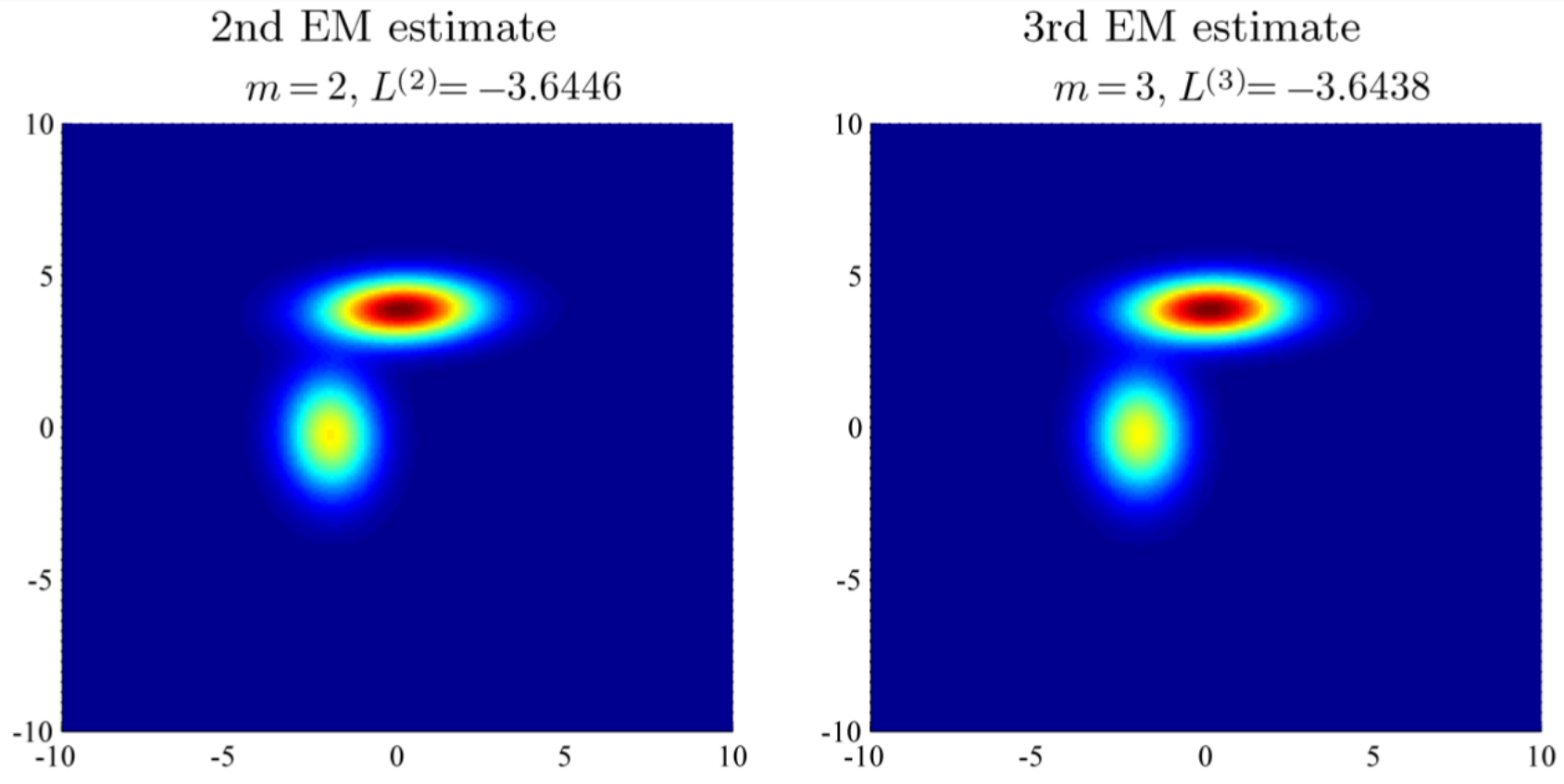
Пример


True GMM density
1000 i.i.d. samples
10
5
0
-5
-10
-10
-5
0
5
10
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.

Суть алгоритма k–средних

- Один из самых известных алгоритмов кластеризации – алгоритм k-средних – это фактически упрощение алгоритма EM.
- Разница в том, что мы не считаем вероятности принадлежности кластерам, а жестко приписываем каждый объект одному кластеру.
- Кроме того, в алгоритме k–средних форма кластеров не настраивается (но это не так важно).

Цель

- Цель алгоритма k–средних — минимизировать меру ошибки

$$E(X, C) = \sum_{i=1}^{n} ||x_i - \mu_i||^2,$$

  где μ_i — ближайший к x_i 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($X, |C|$):

- Инициализировать центры $|C|$ кластеров $\mu_1, \ldots, \mu_{|C|}$.
- Пока принадлежность кластерам не перестанет изменяться:
  - Определить принадлежность x_i к кластерам:

$$\mathrm{clust}_i := \arg\min_{c \in C} \rho(x_i, \mu_c).$$

  - Определить новое положение центров кластеров:

$$\mu_c := \frac{\sum_{\mathrm{clust}_i = c} f_j(x_i)}{\sum_{\mathrm{clust}_i = c} 1}.$$

Semi–supervised clustering

- 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?

Semi–supervised clustering

- Чтобы учесть информацию о точке x_i, достаточно для EM положить скрытую переменную g_{ic} 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 EM: presence-only data

Presence-only data

- Пример из экологии: пусть мы хотим оценить, где водятся те или иные животные.
- Как определить, что суслики тут водятся, понятно: видишь суслика — значит, он есть.
- Но как определить, что суслика нет? Может быть, ты не видишь суслика, и я не вижу, а он есть?..

Presence-only data

- Формально говоря, есть переменные $\mathbf{x}$, определяющие некий регион (квадрат на карте), и мы моделируем вероятность того, что нужный вид тут есть, $p(y = 1 \mid \mathbf{x})$, при помощи логит-функции:

$$p(y = 1 \mid \mathbf{x}) = \sigma(\eta(\mathbf{x})) = \frac{1}{1 + e^{-\eta(\mathbf{x})}},$$

  где $\eta(\mathbf{x})$ может быть линейной (тогда получится логистическая регрессия), но может, в принципе, и не быть.
- Заметим, что даже если бы мы знали настоящие y, это было бы ещё не всё: сэмплирование положительных и отрицательных примеров неравномерно, перекошено в пользу положительных.

Presence-only data

- Это значит, что ещё есть пропорции сэмплирования (sampling rates)

$$\gamma_0 = p(s = 1 \mid y = 0), \quad \gamma_1 = p(s = 1 \mid y = 1),$$

  т.е. вероятности взять в выборку положительный/отрицательный пример.

- Их можно оценить как

$$\gamma_0 = \frac{n_0}{(1-\pi)N}, \quad \gamma_1 = \frac{n_1}{\pi N},$$

  где π — истинная доля положительных примеров (встречаемость, occurrence).

- Кстати, эту π было бы очень неплохо оценить в итоге.

Presence-only data

- Тогда, если знать истинные значения всех y, то в принципе можно обучить:

$$p(y = 1 \mid s = 1, \mathbf{x}) =$$

$$= \frac{p(s = 1 \mid y = 1, \mathbf{x})p(y = 1 \mid \mathbf{x})}{p(s = 1 \mid y = 0, \mathbf{x})p(y = 0 \mid \mathbf{x}) + p(s = 1 \mid y = 1, \mathbf{x})p(y = 1 \mid \mathbf{x})} =$$

$$= \frac{\gamma_1 e^{\eta(\mathbf{x})}}{\gamma_0 + \gamma_1 e^{\eta(\mathbf{x})}} = \frac{e^{\eta^*(\mathbf{x})}}{1 + e^{\eta^*(\mathbf{x})}},$$

где $\eta^*(\mathbf{x}) = \eta(\mathbf{x}) + \log(\gamma_1/\gamma_0)$, т.е.

$$\eta^*(\mathbf{x}) = \eta(\mathbf{x}) + \log\left(\frac{n_1}{n_0}\right) - \log\left(\frac{\pi}{1-\pi}\right).$$

Presence-only data

- Таким образом, если π неизвестно, то $\eta(\mathbf{x})$ можно найти с точностью до константы.
- А у нас не n_0 и n_1, а naive presence n_p и background n_u, т.е.

$$p(y=1 \mid s=1) = \frac{n_p+\pi n_u}{n_p+n_u}, \quad p(y=1 \mid s=0) = \frac{(1-\pi)n_u}{n_p+n_u},$$
$$\gamma_1 = \frac{p(y=1|s=1)p(s=1)}{p(y=1)} = \frac{n_p+\pi n_u}{\pi(n_p+n_u)}p(s=1),$$
$$\gamma_0 = \frac{p(y=0|s=1)p(s=1)}{p(y=0)} = \frac{n_u}{n_p+n_u}p(s=1),$$

  и в нашей модели $\log \frac{n_1}{n_0} = \log \frac{n_p+\pi n_u}{\pi n_u}$, т.е. всё как раньше, но $n_1 = n_p + \pi n_u$, $n_0 = (1-\pi)n_u$.
- Обучать по y можно так: обучить модель, а потом вычесть из $\eta(\mathbf{x})$ константу $\log \frac{n_p+\pi n_u}{\pi n_u}$.

Presence-only data

- Но у нас нет настоящих данных y, чтобы обучить регрессию, а есть только presence-only z: если $z = 1$, то $y = 1$, но если $z = 0$, то неизвестно, чему равен y.
- (Ward et al., 2009): давайте использовать EM. Правдоподобие:

$$\mathcal{L}(\eta \mid \mathbf{y}, \mathbf{z}, \mathcal{X}) = \prod_i p(y_i, z_i \mid s_i = 1, \mathbf{x}_i) = \\ = \prod_i p(y_i \mid s_i = 1, \mathbf{x}_i) p(z_i \mid y_i, s_i = 1, \mathbf{x}_i).$$

- В нашем случае при n_p положительных примеров и n_u фоновых (неизвестных)

$$\begin{aligned} p(z_i = 0 \mid y_i = 0, s_i = 1, \mathbf{x}_i) &= 1, \\ p(z_i = 1 \mid y_i = 1, s_i = 1, \mathbf{x}_i) &= \frac{n_p}{n_p + \pi n_u}, \\ p(z_i = 0 \mid y_i = 1, s_i = 1, \mathbf{x}_i) &= \frac{\pi n_u}{n_p + \pi n_u}. \end{aligned}$$

Presence-only data

- А максимизировать нам надо сложное правдоподобие, в котором значения $\mathbf{y}$ неизвестны:

$$L(\eta \mid \mathbf{z}, \mathcal{X}) = \prod_i p(z_i \mid s_i = 1, \mathbf{x}) =$$

$$= \prod_i \left(\frac{\frac{n_p}{\pi n_u} e^{\eta(\mathbf{x}_i)}}{1 + \left(1 + \frac{n_p}{\pi n_u}\right) e^{\eta(\mathbf{x}_i)}} \right)^{z_i} \left(\frac{1 + e^{\eta(\mathbf{x}_i)}}{1 + \left(1 + \frac{n_p}{\pi n_u}\right) e^{\eta(\mathbf{x}_i)}} \right)^{1 - z_i}.$$

- Для этого и нужен EM.

Presence-only data

- E-шаг здесь в том, чтобы заменить y_i на его оценку

$$\hat{y}_i^{(k)} = \mathbb{E}\left[y_i \mid \eta^{(k)}\right] = \frac{e^{\eta^{(k)}} + 1}{1 + e^{\eta^{(k)}} + 1}.$$

- M-шаг мы уже видели, это обучение параметров логистической модели с целевой переменной $\mathbf{y}^{(k)}$ на данных $\mathcal{X}$.

(1) Chose initial estimates: $\hat{y}_i^{(0)} = \pi$ for $z_i = 0$.
(2) Repeat until convergence:

- *Maximization step:*
  - Calculate $\hat{\eta}^{*(k)}$ by fitting a logistic model of $\hat{\mathbf{y}}^{(k-1)}$ given X.
  - Calculate $\hat{\eta}^{(k)} = \hat{\eta}^{*(k)} - \log\left(\frac{n_p + \pi n_u}{\pi n_u}\right)$.
- *Expectation step:*

$$\hat{y}_i^{(k)} = \frac{e^{\hat{\eta}^{(k)}}}{1 + e^{\hat{\eta}^{(k)}}} \text{ for } z_i = 0 \quad \text{and} \quad \hat{y}_i^{(k)} = 1 \text{ for } z_i = 1$$

Presence-only data

- У Ward et al. получалось хорошо, но тут вышел любопытный спор.
- Ward et al. писали так: хотелось бы, чтобы можно было оценить π, но "π is identifiable only if we make unrealistic assumptions about the structure of $\eta(\mathbf{x})$ such as in logistic regression where $\eta(\mathbf{x})$ is linear in $\mathbf{x}$: $\eta(\mathbf{x}) = \mathbf{x}^\top \boldsymbol{\beta}$".
- Через пару лет вышла статья Royle et al. (2012), тоже очень цитируемая, в которой говорилось: "logistic regression... is hardly unrealistic... such models are the most common approach to modeling binary variables in ecology (and probably all of statistics)... the logistic functions... is customarily adopted and widely used, and even books have been written about it".

Presence-only data

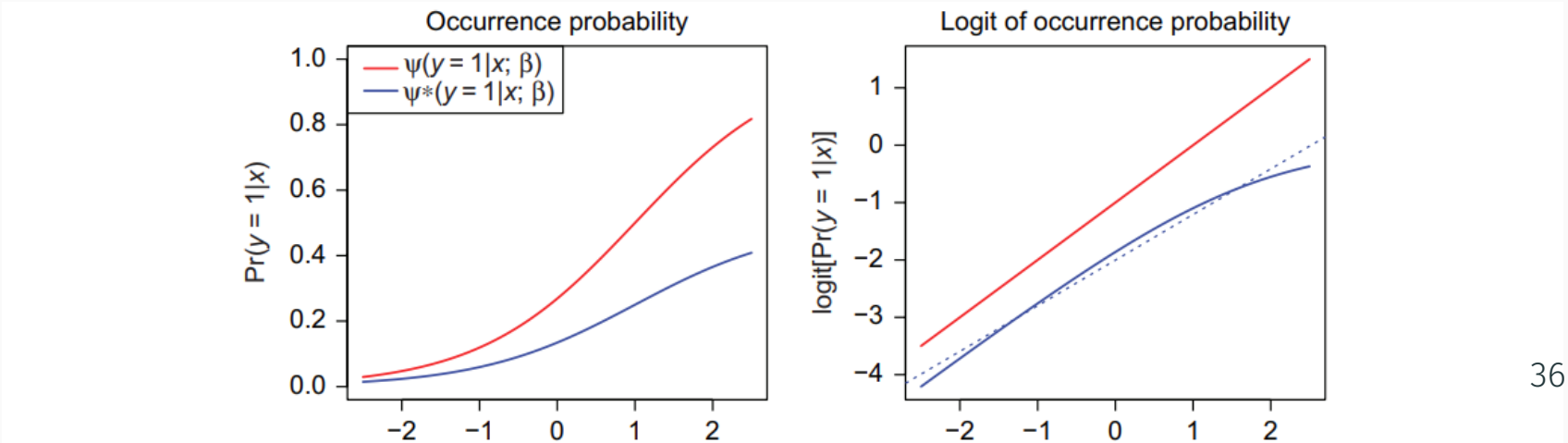
- Они предложили процедуру для оценки встречаемости π:
  - для признаков $\mathbf{x}$ у нас $p(y = 1 \mid \mathbf{x}) = \frac{p(y=1)\pi_1(\mathbf{x})}{p(y=1)\pi_1(\mathbf{x})+(1-p(y=1))\pi_0(\mathbf{x})}$;
  - данные – это выборка из $\pi_1(\mathbf{x})$ и отдельно выборка из $\pi(\mathbf{x})$;
  - как видно, даже если знать π и π_1 полностью, остаётся свобода: надо оценить $p(y = 1 \mid \mathbf{x})$, а $\pi_0(\mathbf{x})$ мы не знаем;
  - Royle et al. вводят предположения для $p(y = 1 \mid \mathbf{x})$ в виде логистической регрессии: $p(y = 1 \mid \mathbf{x}) = \sigma(\boldsymbol{\beta}^\top \mathbf{x})$;
  - тогда действительно можно записать $p(y = 1)\pi_1(\mathbf{x}) = p(y = 1 \mid \mathbf{x})\pi(\mathbf{x}) = p(y = 1, \mathbf{x})$, и если $\pi(\mathbf{x})$ равномерно (это логично), то

$$\pi_1(\mathbf{x}_i) = \frac{p(y_i = 1 \mid \mathbf{x}_i)}{\sum_{\mathbf{x}} p(y = 1 \mid \mathbf{x})};$$

    если подставить сюда логистическую регрессию, то можно оценить $\boldsymbol{\beta}$ максимального правдоподобия, и не надо знать $p(y = 1)$!
- Что тут не так?

Presence-only data

- Всё так, но предположение о логистической регрессии здесь выполняет слишком много работы.
- Если рассмотреть две кривые, у которых $p^*(y = 1 \mid \mathbf{x}, \boldsymbol{\beta}) = \frac{1}{2} p(y = 1 \mid \mathbf{x}, \boldsymbol{\beta})$, то у них будет $p^*(y = 1) = \frac{1}{2} p(y = 1)$, но общее правдоподобие $\pi_1(\mathbf{x}_i)$ будет в точности одинаковое, $\frac{1}{2}$ сократится.
- Дело в том, что модель p^* не будет логистической регрессией, с $\boldsymbol{\beta}$ произойдёт что-то нелинейное; но откуда у нас настолько сильное предположение? Как отличить синюю кривую справа от пунктирной прямой?



Рейтинг спортивного ЧГК

- Пример из практики: в какой-то момент я хотел сделать рейтинг спортивного «Что? Где? Когда?»:
  - участвуют команды по ≤ 6 человек, причём часто встречаются неполные команды;
  - игроки постоянно переходят между командами (поэтому TrueSkill);
  - в одном турнире могут участвовать до тысячи команд (синхронные турниры);
  - командам задаётся фиксированное число вопросов (36, 60, 90), т.е. в крупных турнирах очень много команд делят одно и то же место.

Рейтинг спортивного ЧГК

- Первое решение — система TrueSkill
- Расскажу о ней потом, когда будем говорить об Expectation Propagation
- У неё есть проблемы в постановке спортивного ЧГК, мы когда-то их отчасти решили, была сложная и интересная модель, она работала лучше базового TrueSkill
- Но...

Рейтинг спортивного ЧГК

- В какой-то момент база турниров ЧГК стала собирать повопросные результаты.
- Мы теперь знаем, на какие именно вопросы ответила та или иная команда.
- Так что когда я вернулся к задаче построения рейтинга ЧГК, задача стала существенно проще.

Рейтинг спортивного ЧГК

- Пример аналогичного приложения:
  - есть набор вопросов для теста из большого числа вопросов (например, IQ-тест или экзамен по какому-то предмету);
  - участники отвечают на случайное подмножество вопросов;
  - надо оценить участников, но уровень сложности вопросов нельзя заранее точно сбалансировать.
- «Что? Где? Когда?» — это оно и есть, только теперь участники объединяются в команды.

Baseline: логистическая регрессия

- Baseline — логистическая регрессия:
  - каждый игрок i моделируется скиллом s_i,
  - каждый вопрос q моделируется сложностью («лёгкостью») c_q,
  - добавим глобальное среднее μ,
  - обучим логистическую модель

$$p(x_{tq} \mid s_i, c_q) \sim \sigma(\mu + s_i + c_q)$$

для каждого игрока $i \in t$ команды-участницы $t \in \mathcal{T}^{(d)}$ и каждого вопроса $q \in Q^{(d)}$, где $\sigma(x) = 1/(1 + e^x)$ — логистический сигмоид, x_{tq} — ответила ли команда t на вопрос q.

Модель со скрытыми переменными

- Логистическая модель предполагает фактически, что каждый игрок ответил на каждый вопрос, который взяла команда.
- Это неправда, мы не знаем, кто ответил, только знаем, что кто-то это сделал; а если команда не ответила, то никто.
- Это по идее похоже на presence-only data models (Ward et al., 2009; Royle et al., 2012).

Модель со скрытыми переменными

- Поэтому давайте сделаем модель со скрытыми переменными.
- Для каждой пары игрок-вопрос, добавим переменную z_{iq}, которая означает, что «игрок i ответил на вопрос q».
- На эти переменные есть такие ограничения:
  - если $x_{tq} = 0$, то $z_{iq} = 0$ для каждого игрока $i \in t$;
  - если $x_{tq} = 1$, то $z_{iq} = 1$ для по крайней мере одного игрока $i \in t$.

Модель со скрытыми переменными

- Параметры модели те же — скилл и сложность вопросов:

$$p(z_{iq} \mid s_i, c_q) \sim \sigma(\mu + s_i + c_q).$$

- Обучаем EM-алгоритмом:
  - E-шаг: зафиксируем все s_i и c_q, вычислим ожидания скрытых переменных z_{iq} как

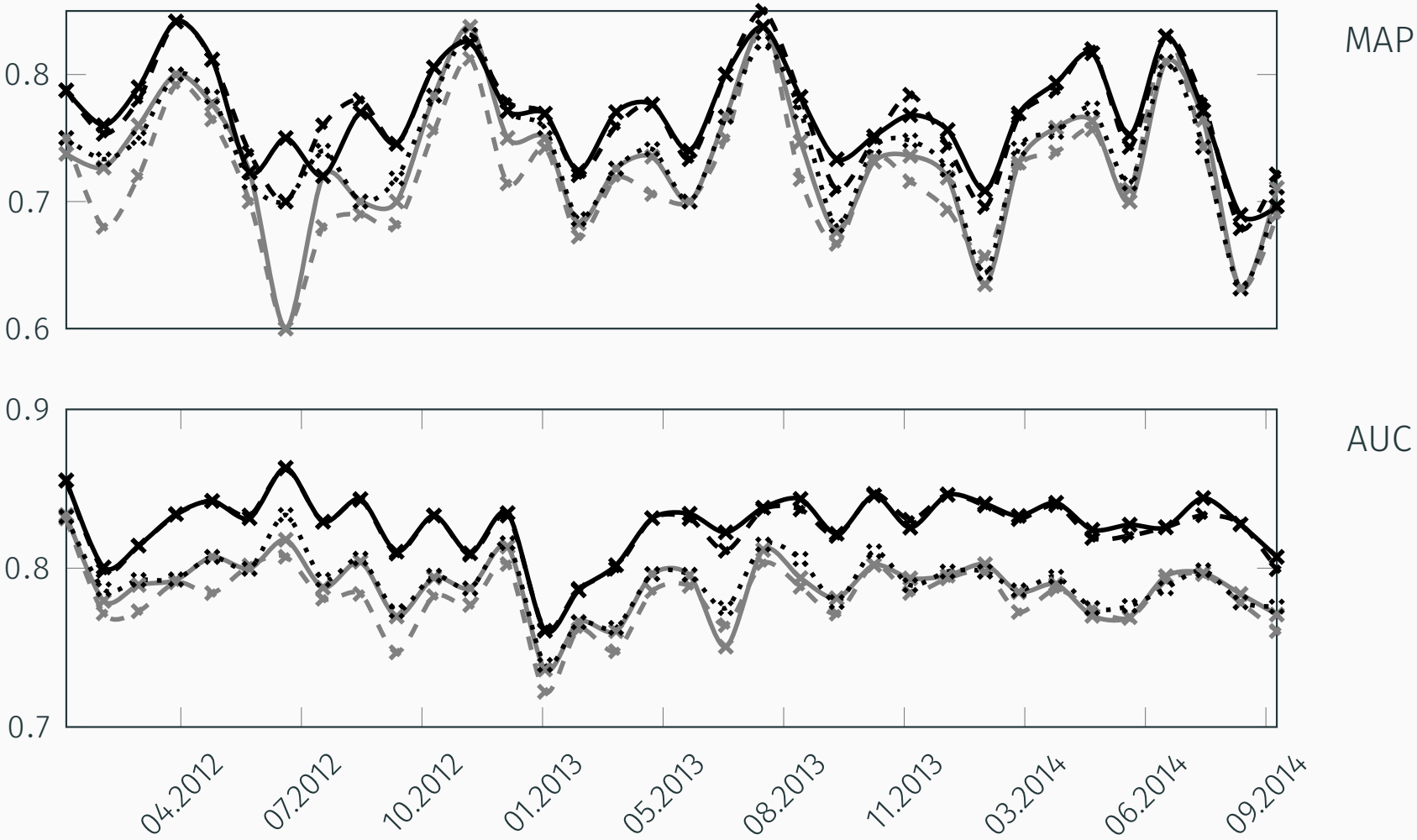
$$\mathbb{E}\left[z_{iq}\right] = \begin{cases} 0, & \text{если } x_{tq} = 0, \\ p(z_{iq} = 1 \mid \exists j \in t\; z_{jq} = 1) = \frac{\sigma(\mu+s_i+c_q)}{1-\prod_{j\in t}\left(1-\sigma(\mu+s_j+c_q)\right)}, & \text{если } x_{tq} = 1; \end{cases}$$

  - M-шаг: зафиксируем $\mathbb{E}\left[z_{iq}\right]$, обучим логистическую модель

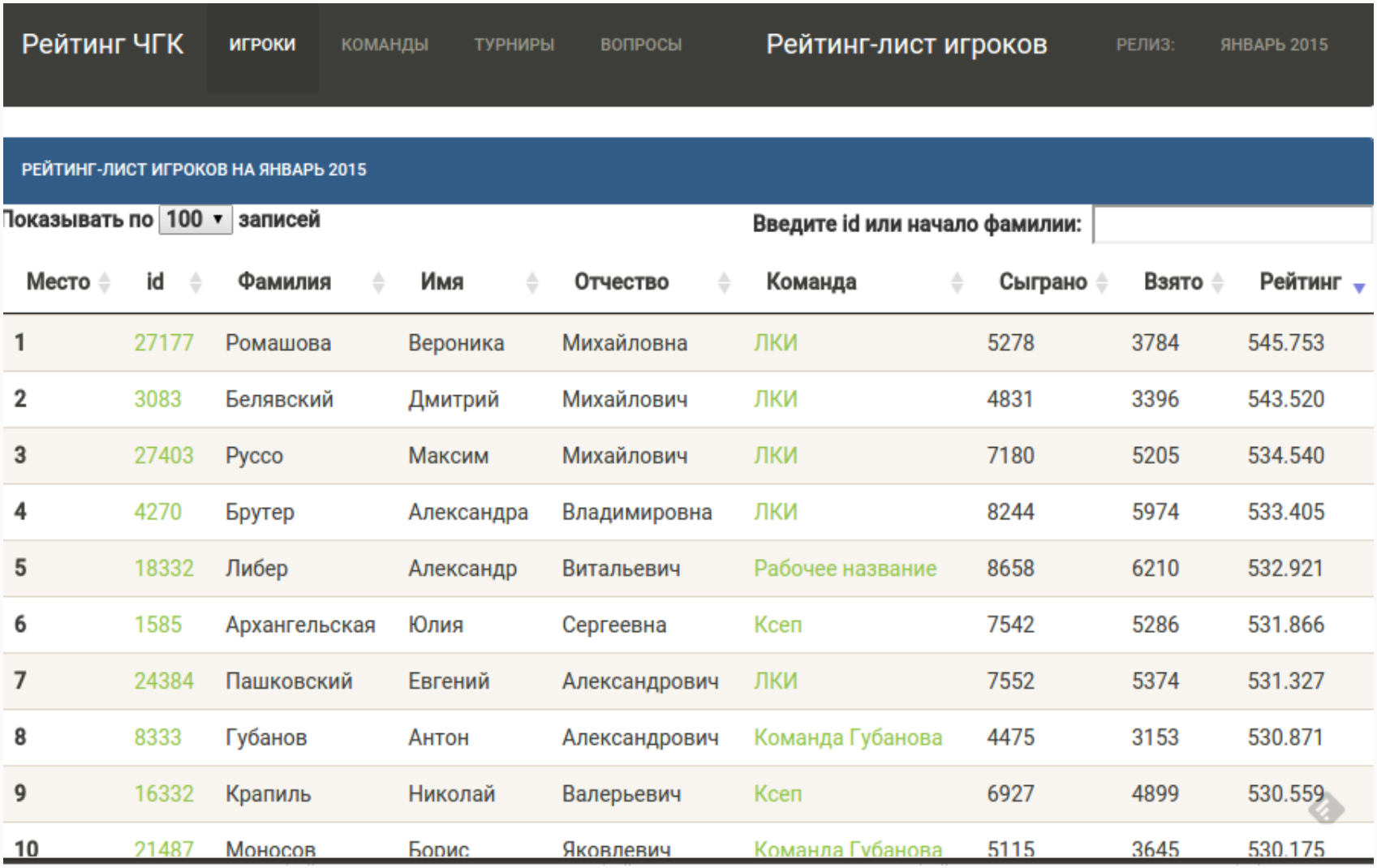
$$\mathbb{E}\left[z_{iq}\right] \sim \sigma(\mu + s_i + c_q).$$

Результаты

- Ну и, конечно, работает хорошо.



Пример



Рейтинг ЧГК | ИГРОКИ | КОМАНДЫ | ТУРНИРЫ | ВОПРОСЫ | Рейтинг-лист игроков | РЕЛИЗ: ЯНВАРЬ 2015

РЕЙТИНГ-ЛИСТ ИГРОКОВ НА ЯНВАРЬ 2015

Показывать по 100 записей

Введите id или начало фамилии:

Место	id	Фамилия	Имя	Отчество	Команда	Сыграно	Взято	Рейтинг
1	27177	Ромашова	Вероника	Михайловна	ЛКИ	5278	3784	545.753
2	3083	Белявский	Дмитрий	Михайлович	ЛКИ	4831	3396	543.520
3	27403	Руссо	Максим	Михайлович	ЛКИ	7180	5205	534.540
4	4270	Брутер	Александра	Владимировна	ЛКИ	8244	5974	533.405
5	18332	Либер	Александр	Витальевич	Рабочее название	8658	6210	532.921
6	1585	Архангельская	Юлия	Сергеевна	Ксеп	7542	5286	531.866
7	24384	Пашковский	Евгений	Александрович	ЛКИ	7552	5374	531.327
8	8333	Губанов	Антон	Александрович	Команда Губанова	4475	3153	530.871
9	16332	Крапиль	Николай	Валерьевич	Ксеп	6927	4899	530.559
10	21487	Моносов	Борис	Яковлевич	Команда Губанова	5115	3645	530.175

Пример

Рейтинг ЧГК ИГРОКИ КОМАНДЫ ТУРНИРЫ ВОПРОСЫ Александр Друзь vs. Максим Поташев

Ссылка на сайт рейтинга МАК

Друзь
Александр Абрамович

Поташев
Максим Оскарович

Ссылка на сайт рейтинга МАК

ИСТОРИЯ РЕЙТИНГА | ИСТОРИЯ ТУРНИРОВ

ИСТОРИЯ РЕЙТИНГОВ

Рейтинг	Команда	Место	Дата	Место	Команда	Рейтинг
450.138	Транссфера	303	Январь 2015	31	Афина	514.643
446.669	Транссфера	336	Декабрь 2014	29	Афина	514.697
442.559	Транссфера	348	Ноябрь 2014	25	Афина	513.155
437.999	Транссфера	400	Октябрь 2014	25	Афина	512.083
445.791	Транссфера	306	Сентябрь 2014	23	Афина	515.737
441.473	Транссфера	372	Август 2014	23	Афина	516.513
446.730	Транссфера	284	Июль 2014	22	Афина	516.755
448.598	Транссфера	322	Июнь 2014	24	Афина	516.603

Спасибо!

Спасибо за внимание!