# КЛАСТЕРИЗАЦИЯ

Сергей Николенко

СПбГУ — Санкт-Петербург
9 ноября 2019 г.

*Random facts*:

- 9 ноября в Украине — День украинской письменности и языка, а в России — День специального отряда быстрого реагирования
- 9 ноября в православной церкви — день памяти преподобного Нестора Летописца; черноризец Печерского монастыря упоминается только в одном, Хлебниковском списке «Повести временных лет», но в целом сомнений в его существовании нет
- 9 ноября — день смены власти: в 1520 г. Кристиан II в Стокгольме расправился над своими противниками, в 1799 г. Наполеон ликвидировал Директорию (18 брюмера), в 1900 г. российская армия завершила оккупацию Маньчжурии, в 1918 Вильгельм II отрёкся от престола в ходе Ноябрьской революции, в 1923 был подавлен Пивной путч в Мюнхене, а в 1989 и Берлинскую стену начали разрушать, и Дэн Сяопин ушёл в отставку
- 9 ноября 1927 г. в Китае обнаружили гигантскую панду
- 9 ноября 1979 г. в компьютер системы NORAD (North American Aerospace Defense Command) загрузили не ту ленту, и в США была объявлена ядерная тревога, а мир в течение десяти минут находился на краю ядерной войны

# 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 $\rho$.
- 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.

# 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 $\mathrm{CF}_i$, $\mathrm{LS}_i = \sum_{\mathbf{x} \in \mathrm{CF}_i} x_i$ (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: $\mathrm{CF}_i$ аддитивны.

# Birch

- CF-дерево состоит из $\mathrm{CF}_i$; 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 $\mu_1$, $\mu_2$.

# 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 $X = \{x_1, \dots, x_N\}$.

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

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

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

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

# Обоснование алгоритма EM

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

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

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

# Обоснование алгоритма EM

- E-шаг алгоритма EM вычисляет условное ожидание (логарифма) полного правдоподобия при условии $X$ и текущих оценок параметров $\theta$:

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

- Условное ожидание – это

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

  где $p(y \mid X, \theta_n)$ – маргинальное распределение скрытых компонентов данных.
- EM лучше всего применять, когда это выражение легко подсчитать, может быть, даже аналитически.
- Вместо $p(y \mid X, \theta_n)$ можно подставить $p(y, X \mid \theta_n) = p(y \mid X, \theta_n) p(X \mid \theta_n)$, 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 $\theta_{n+1}$, 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 $\theta_n$ к $\theta$, для которого $\ell(\theta) > \ell(\theta_n)$.

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

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

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

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

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

# Обоснование алгоритма EM

- Получили

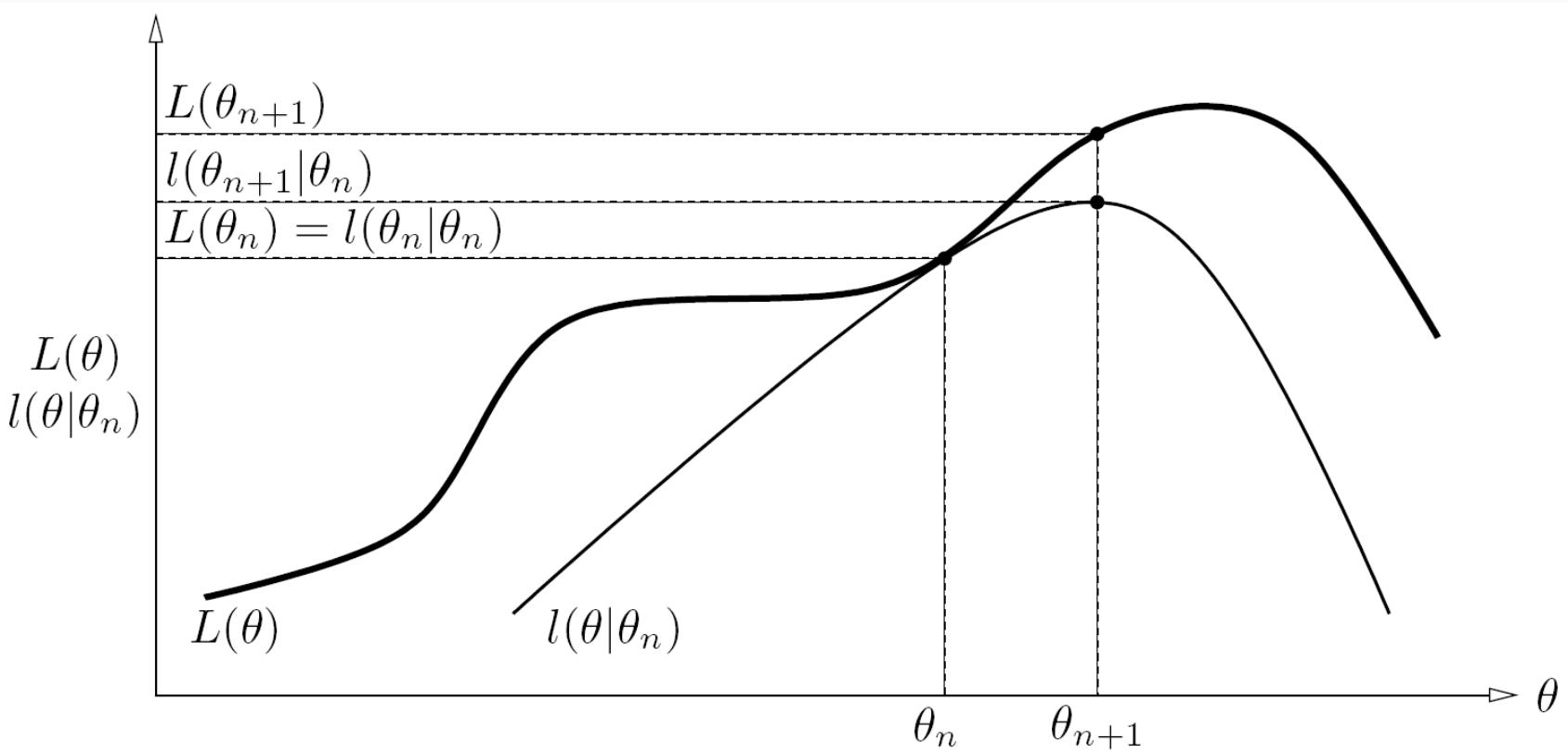
$$\ell(\theta) \geq l(\theta, \theta_n) =$$
$$= \ell(\theta_n) + \int_y p(y \mid X, \theta_n) \log \left( \frac{p(X \mid y, \theta) p(y \mid \theta)}{p(X \mid \theta_n) p(y \mid X, \theta_n)} \right) \mathrm{d}y.$$

**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 $\theta_n$.
- 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 $Q$.

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

а остальное от $\theta$ 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 $\mu_c = \{\mu_{c1}, \dots, \mu_{cd}\}$ 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 $(f_1(x), \ldots, f_n(x))$.
- 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 $w$, $\mu$, $\sigma$:

# 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 $w$, $\mu$, $\sigma$:

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

# 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 $w$, $\mu$, $\sigma$:

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

# 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 $w$, $\mu$, $\sigma$:

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

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

# Алгоритм

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

- Инициализировать $|C|$ кластеров; начальное приближение:
  $w_c := 1/|C|$, $\mu_c :=$ случайный $x_i$,
  $\sigma^2_{cj} := \frac{1}{n|C|}\sum_{i=1}^{n}\big(f_j(x_i) - \mu_{cj}\big)^2$.
- 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: $w_c = \frac{1}{n}\sum_{i=1}^{n} g_{ic}$, $\mu_{cj} = \frac{1}{nw_c}\sum_{i=1}^{n} g_{ic} f_j(x_i)$,

$$\sigma^2_{cj} = \frac{1}{nw_c}\sum_{i=1}^{n} g_{ic}\big(f_j(x_i) - \mu_{cj}\big)^2.$$

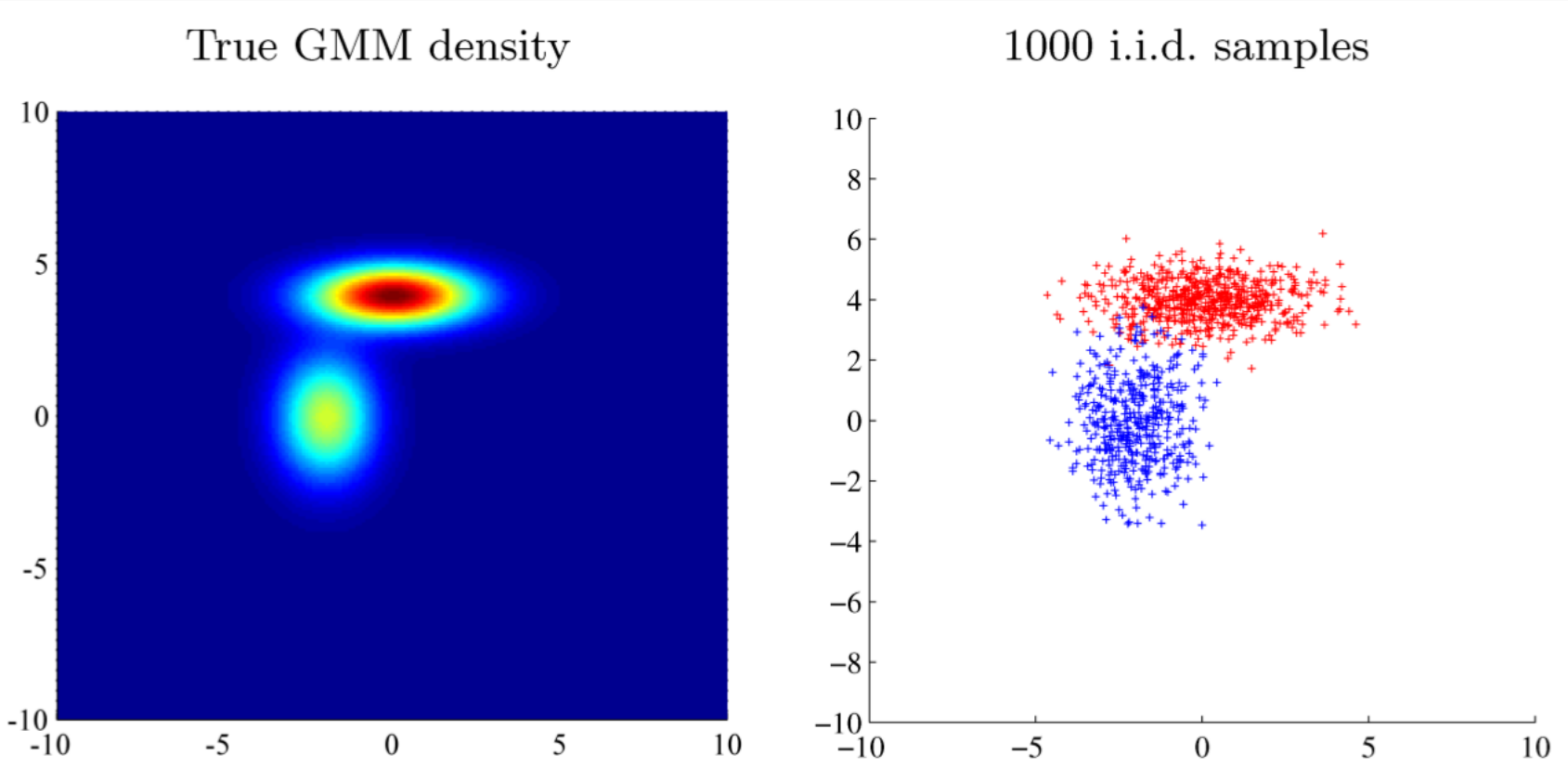
  - Определить принадлежность $x_i$ к кластерам:

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

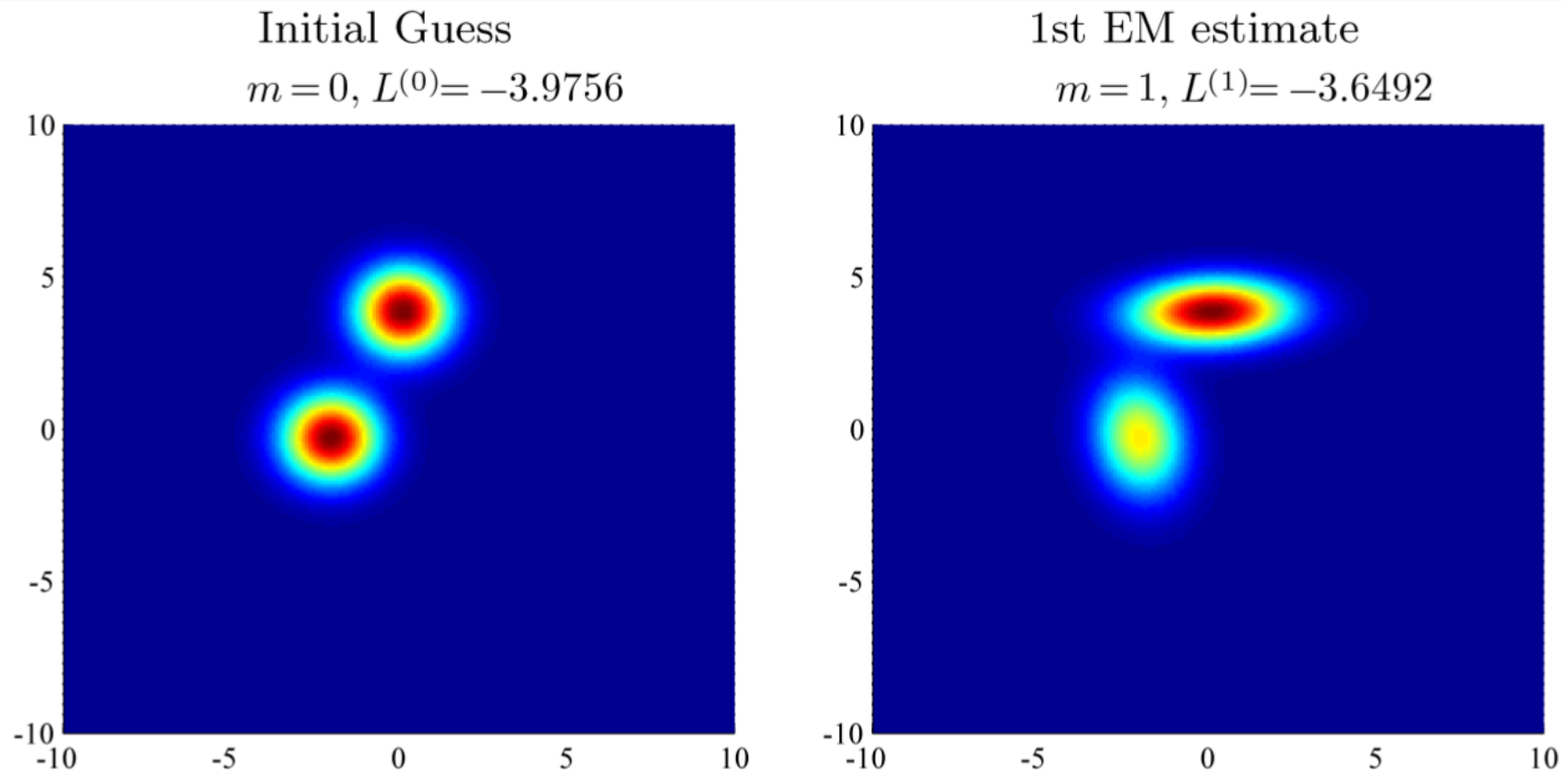
# Упражнение

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

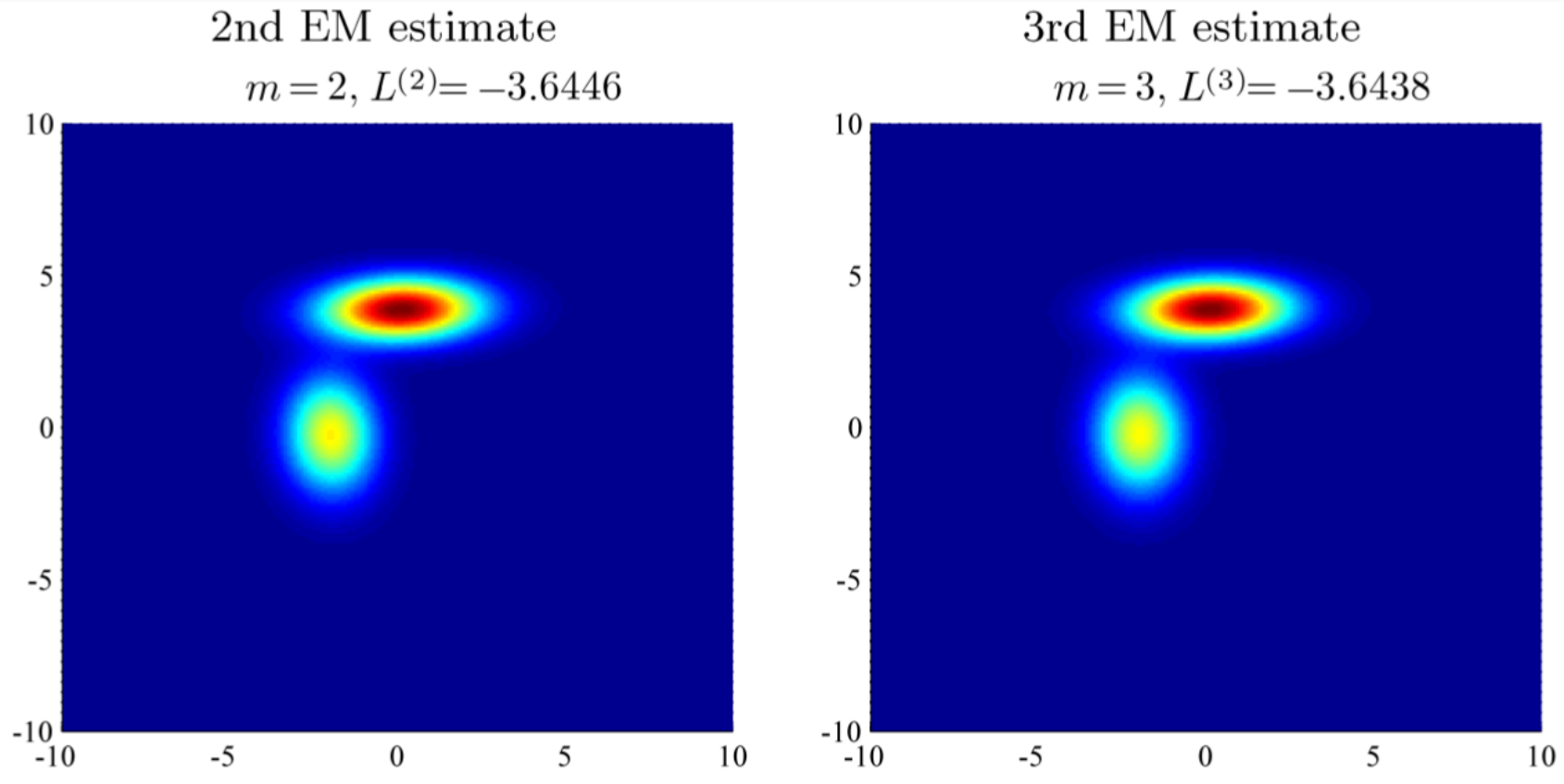
# Пример


True GMM density
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.

# 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 $\mu_i$ — 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 $\mu_1, \dots, \mu_{|C|}$.
- Пока принадлежность кластерам не перестанет изменяться:
  - Определить принадлежность $x_i$ к кластерам:

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

  - Определить новое положение центров кластеров:

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

# 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 $\mathrm{clust}_i$.

# Объединение моделей

# Как объединять модели

- До сих пор мы разрабатывали (и потом ещё будем разрабатывать) модели, которые делают предсказания (в основном для задач регрессии и классификации).
- Таким образом, мы можем попробовать обучить сразу много разных моделей!
- Model selection – это о том, как выбрать из них лучшую.
- Но, может быть, можно не выбирать, а использовать все сразу?

# Основные подходы

- Комитет: обучаем $L$ разных моделей, а потом так или иначе усредняем-комбинируем их результаты.
- Альтернатива: обучаем $L$ разных моделей, а потом обучаем отдельную модель о том, какую из них использовать для предсказания (например, дерево принятия решений).

# Комбинация моделей и байесовское усреднение

- Начнём с самого простого – байесовского усреднения.
- Мы уже знаем, что такое комбинация моделей – например, линейная смесь гауссианов:

$$p(\mathbf{x}) = \sum_k \pi_k N(\mathbf{x} \mid \mu_k, \Sigma_k).$$

- Если её обучать, мы обучим коэффициенты смеси, и результат будет порождён как бы двухуровневым процессом.

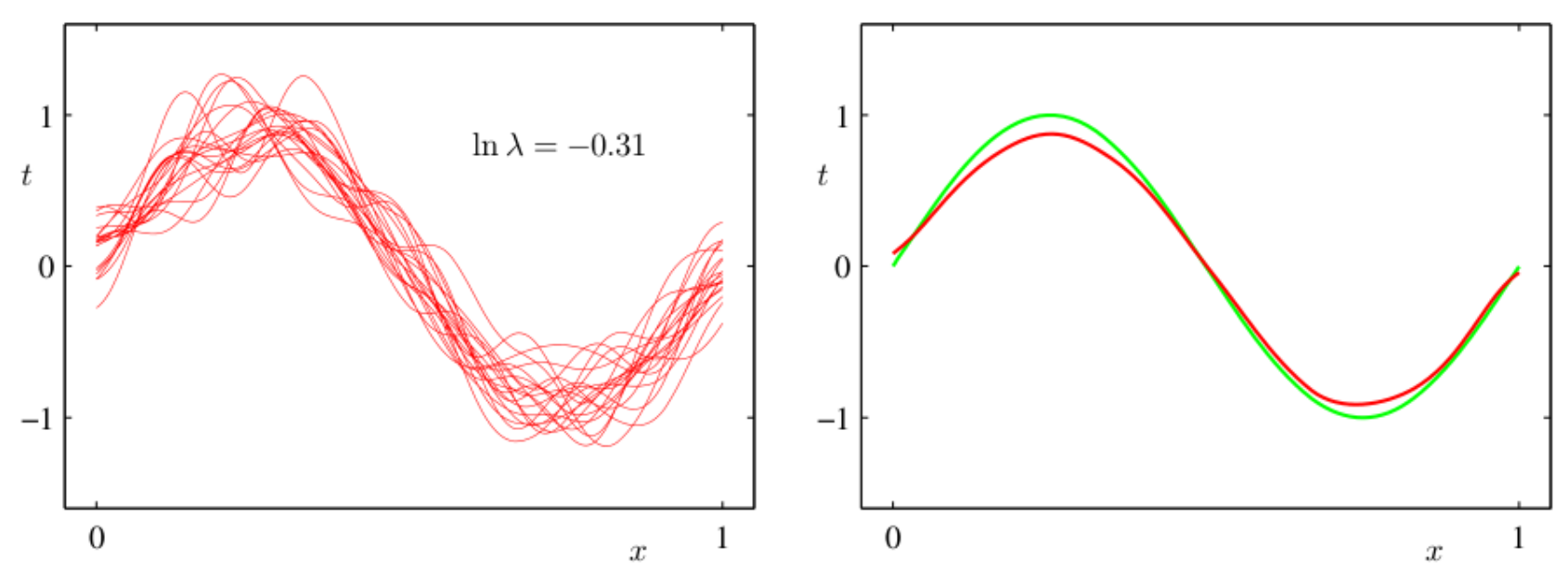
# Комбинация моделей и байесовское усреднение

- А байесовское усреднение будет выглядеть как

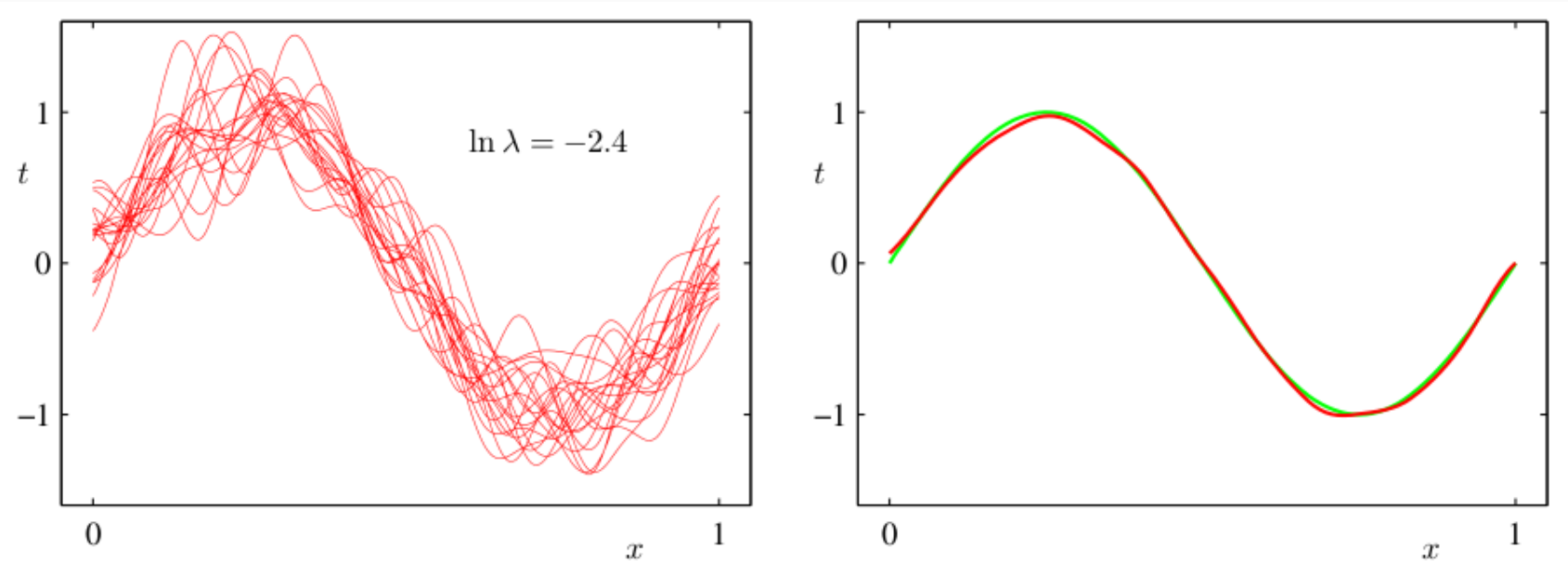
$$p(\mathbf{X}) = \sum_{h=1}^{H} p(\mathbf{X} \mid h) p(h).$$

- Смысл теперь в том, что генерирует $\mathbf{X}$ только одна модель, но мы просто не знаем какая именно; когда $\mathbf{X}$, апостериорные распределения $p(h \mid \mathbf{X})$ сужаются, и мы выбираем то, что надо.
- Но метод очень простой: взять много моделей и усреднить.
- Где-то мы это уже видели...

# Где-то мы это уже видели



# Где-то мы это уже видели



# Bagging

- На этих картинках – модели с высоким bias, которые обучены по разным датасетам, сгенерированным одним и тем же распределением.
- И если их усреднить, получится как раз то, что надо.
- Но в жизни у нас нет возможности генерировать много датасетов: сколько данных есть, столько есть.
- Просто разбивать датасет на части – не поможет. Что делать?

# Bagging

- Пусть у нас есть датасет $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$.
- Давайте сгенерируем много датасетов так: будем выбирать из $\mathbf{X}$ $N$ точек с замещением, т.е. в новом датасете некоторые точки будут повторяться.
- Этот метод называется *bootstrapping*.

# Bagging

- Мы сделаем так $M$ датасетов размера $N$ (с повторяющимися точками), потом обучим $M$ моделей, а потом образуем из них комитет и будем предсказывать как

$$y(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^{M} y_m(\mathbf{x}).$$

- Это называется *bagging* (bootstrap aggregation).
- На первый взгляд кажется, что это какая-то ерунда: мы пытаемся получить что-то из ничего...

# Bagging

- Пусть настоящая функция, которую мы пытаемся предсказать – $h(\mathbf{x})$, т.е. модели наши выглядят как

$$y_m(\mathbf{x}) = h(\mathbf{x}) + \epsilon_m(\mathbf{x}).$$

- Тогда средняя ошибка модели – это

$$\mathrm{E}_{\mathbf{x}}\left[\left(y_m(\mathbf{x}) - h(\mathbf{x})\right)^2\right] = \mathrm{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})^2\right].$$

- И средняя ошибка тех моделей, которые мы обучаем, получается

$$E_{\mathrm{avg}} = \frac{1}{M}\sum_{m=1}^{M} \mathrm{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})^2\right].$$

# Bagging

- И средняя ошибка тех моделей, которые мы обучаем, получается

$$E_{\text{avg}} = \frac{1}{M} \sum_{m=1}^{M} \mathrm{E}_{\mathbf{x}} \left[ \epsilon_m(\mathbf{x})^2 \right].$$

- А ошибка комитета – это

$$E_{\text{com}} = \mathrm{E}_{\mathbf{x}} \left[ \left( \frac{1}{M} \sum_{m=1}^{M} (y_m(\mathbf{x}) - h(\mathbf{x})) \right)^2 \right] =$$

$$= \mathrm{E}_{\mathbf{x}} \left[ \left( \frac{1}{M} \sum_{m=1}^{M} \epsilon_m(\mathbf{x}) \right)^2 \right].$$

# Bagging

- $E_{\text{avg}} = \frac{1}{M} \sum_{m=1}^{M} \mathrm{E}_{\mathbf{x}} \left[ \epsilon_m(\mathbf{x})^2 \right]$, $E_{\text{com}} = \mathrm{E}_{\mathbf{x}} \left[ \left( \frac{1}{M} \sum_{m=1}^{M} \epsilon_m(\mathbf{x}) \right)^2 \right]$.
- Если предположить, что $\mathrm{E}_{\mathbf{x}} \left[ \epsilon_m(\mathbf{x}) \right] = 0$, и ошибки некоррелированы: $\mathrm{E}_{\mathbf{x}} \left[ \epsilon_m(\mathbf{x}) \epsilon_l(\mathbf{x}) \right] = 0$, мы получим

$$E_{\text{com}} = \frac{1}{M} E_{\text{avg}}.$$

# Bagging

- $E_{\text{com}} = \frac{1}{M} E_{\text{avg}}$!
- Это кажется совершенно невероятным. На самом деле всё не так хорошо – конечно, ошибки на самом деле сильно коррелированы.
- И, конечно, на самом деле обычно уменьшение ошибки не такое большое.
- Но можно показать, что в любом случае $E_{\text{com}} \leq E_{\text{avg}}$, так что хуже от этого не будет, а лучше стать может.

# Спасибо!

Спасибо за внимание!