

# КЛАСТЕРИЗАЦИЯ И АЛГОРИТМ EM

---

Сергей Николенко

НИУ ВШЭ — Санкт-Петербург

15 сентября 2018 г.

---

## *Random facts:*

- 15 сентября 1928 г. Александр Флеминг впервые в мире выделил пенициллин
- 15 сентября 1933 г. Ягода сообщил Сталину о раскрытии в Ленинграде «общества педерастов»; были арестованы более 150 человек
- 15 сентября 1974 г. на пересечении улиц Островитянова и Профсоюзной в Беляево работники по благоустройству и развитию лесопарка крайне нервно, с применением тяжелой техники отреагировали на попытку сорвать им субботник
- 15 сентября 1997 г. был зарегистрирован домен google.com
- 15 сентября 2007 г. близ деревни Каранкас около Титикаки упал необычный метеорит; в то же время 600 жителей деревни начали страдать от разных болезней с неясными причинами

# 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  $\rho$ .
- 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.

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

- 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 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?

- 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  $CF_i$ ,  
 $LS_i = \sum_{x \in CF_i} x_i$  (linear sum),  $SS_i = \sum_{x \in CF_i} x_i^2$  (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 И КЛАСТЕРИЗАЦИЯ

---

- 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

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

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  $\mu_1, \mu_2$ .

- 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.

- 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:

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

Мы подсчитываем эти ожидания, а потом подправляем гипотезу:

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

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

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

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

- EM 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  $Z = (X, Y)$  с совместной плотностью

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

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

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

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

- 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  $\theta$ :

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

- Условное ожидание – это

$$E[\log p(X, Y | \theta) | X, \theta_n] = \int_y \log p(X, y | \theta) p(y | X, \theta_n) dy,$$

где  $p(y | X, \theta_n)$  – маргинальное распределение скрытых компонентов данных.

- EM лучше всего применять, когда это выражение легко подсчитать, может быть, даже аналитически.
- Вместо  $p(y | X, \theta_n)$  можно подставить  $p(y, X | \theta_n) = p(y | X, \theta_n)p(X | \theta_n)$ , от этого ничего не изменится.

- 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.

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

$$\begin{aligned}
 \ell(\theta) - \ell(\theta_n) &= \\
 &= \log \left( \int_y p(X | y, \theta) p(y | \theta) dy \right) - \log p(X | \theta_n) = \\
 &= \log \left( \int_y p(y | X, \theta_n) \frac{p(X | y, \theta) p(y | \theta)}{p(y | X, \theta_n)} dy \right) - \log p(X | \theta_n) \geq \\
 &\geq \int_y p(y | X, \theta_n) \log \left( \frac{p(X | y, \theta) p(y | \theta)}{p(y | X, \theta_n)} \right) dy - \log p(X | \theta_n) = \\
 &= \int_y p(y | X, \theta_n) \log \left( \frac{p(X | y, \theta) p(y | \theta)}{p(X | \theta_n) p(y | X, \theta_n)} \right) dy.
 \end{aligned}$$

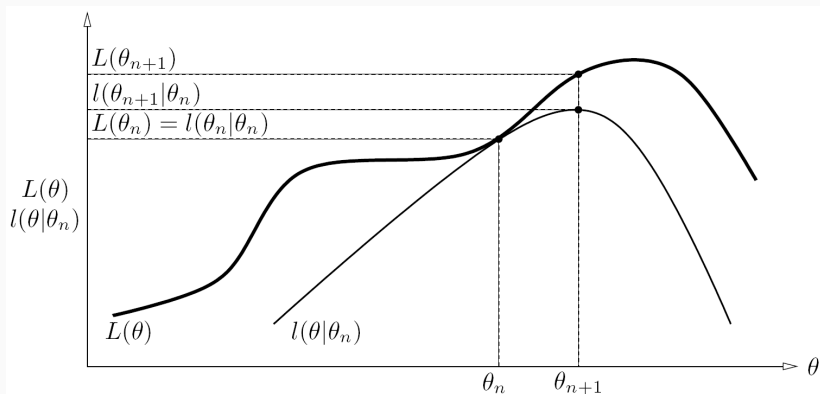
- Получили

$$\begin{aligned}\ell(\theta) &\geq l(\theta, \theta_n) = \\ &= \ell(\theta_n) + \int_y p(y | X, \theta_n) \log \left( \frac{p(X | y, \theta)p(y | \theta)}{p(X | \theta_n)p(y | X, \theta_n)} \right) dy.\end{aligned}$$

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

- 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





- Осталось только понять, что максимизировать можно  $Q$ .

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

а остальное от  $\theta$  не зависит. Вот и получился EM.

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

- 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  $c$ .

- 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.
- 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  $\Sigma_c = \text{diag}(\sigma_{c1}^2, \dots, \sigma_{cd}^2)$  (т.е. по каждой координате своя дисперсия).

- В этих предположениях получается в точности задача разделения смеси вероятностных распределений. Для этого и нужен EM–алгоритм.
- Каждый тестовый пример описывается своими координатами  $(f_1(x), \dots, f_n(x))$ .
- Скрытые переменные в данном случае — вероятности  $g_{ic}$  того, что объект  $x_i$  принадлежит кластеру  $c \in C$ .

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



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

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

- $E$ -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  $w, \mu, \sigma$ :

- *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  $w, \mu, \sigma$ :

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

- *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  $w, \mu, \sigma$ :

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

- *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  $w, \mu, \sigma$ :

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

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

EMCluster( $X, |C|$ ):

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

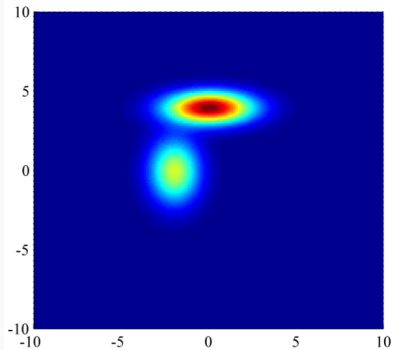
$$\sigma_{cj}^2 = \frac{1}{nw_c} \sum_{i=1}^n g_{ic} (f_j(x_i) - \mu_{cj})^2.$$

- Определить принадлежность  $x_i$  к кластерам:

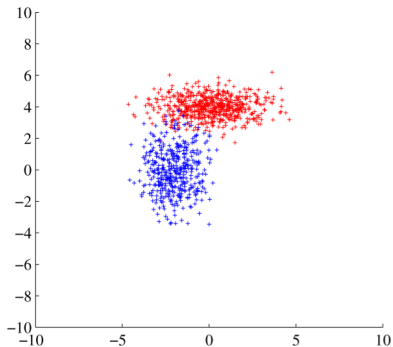
$$\text{clust}_i := \arg \max_{c \in C} g_{ic}.$$

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

True GMM density



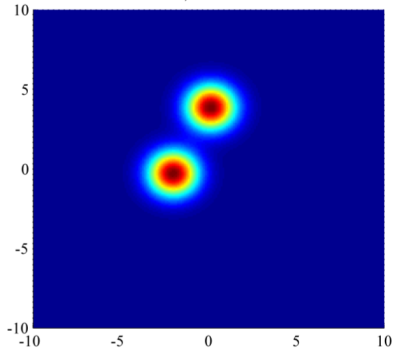
1000 i.i.d. samples





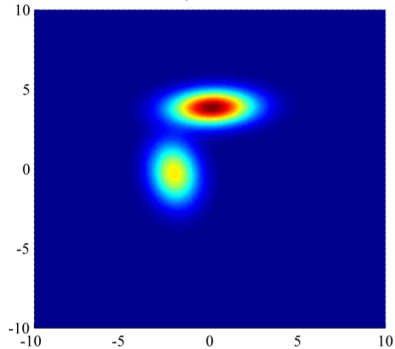
Initial Guess

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



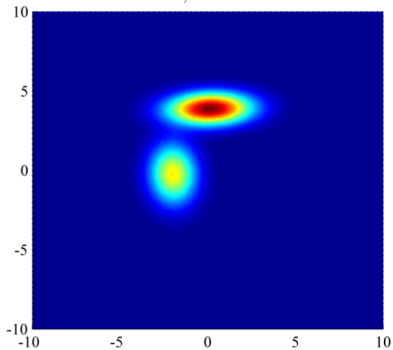
1st EM estimate

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



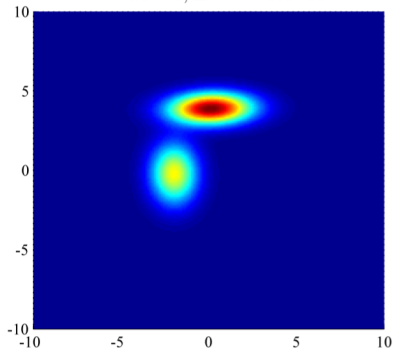
2nd EM estimate

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



3rd EM estimate

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



- Остается проблема: нужно задавать количество кластеров.

- 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 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.

- 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( $X, |C|$ ):

- 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:

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

- 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  $x_i$ , 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  $\text{clust}_i$ .

# СКРЫТЫЕ МАРКОВСКИЕ МОДЕЛИ: ОСНОВНОЕ

---

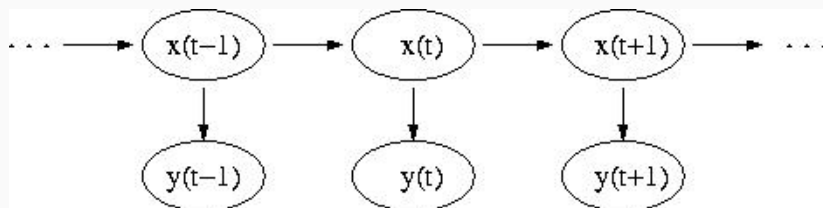
- Марковская цепь задаётся начальным распределением вероятностей  $p^0(x)$  и вероятностями перехода  $T(x'; x)$ .
- $T(x'; x)$  — это распределение следующего элемента цепи в зависимости от следующего; распределение на  $(t + 1)$ -м шаге равно

$$p^{t+1}(x') = \int T(x'; x)p^t(x)dx.$$

- В дискретном случае  $T(x'; x)$  — это матрица вероятностей  $p(x' = i | x = j)$ .

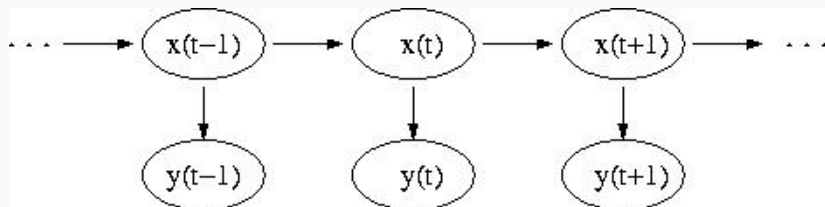
# ДИСКРЕТНЫЕ МАРКОВСКИЕ ЦЕПИ

- Мы будем находиться в дискретном случае.
- Марковская модель — это когда мы можем наблюдать какие-то функции от марковского процесса.



# ДИСКРЕТНЫЕ МАРКОВСКИЕ ЦЕПИ

- Здесь  $x(t)$  — сам процесс (модель), а  $y(t)$  — то, что мы наблюдаем.
- Задача — определить скрытые параметры процесса.



- Главное свойство — следующее состояние не зависит от истории, только от предыдущего состояния.

$$\begin{aligned} p(x(t) = x_j | x(t-1) = x_{j_{t-1}}, \dots, x(1) = x_{j_1}) = \\ = p(x(t) = x_j | x(t-1) = x_{j_{t-1}}). \end{aligned}$$

- Более того, эти вероятности  $a_{ij} = p(x(t) = x_j | x(t-1) = x_i)$  ещё и от времени  $t$  не зависят.
- Эти вероятности и составляют матрицу перехода  $A = (a_{ij})$ .

- Естественные свойства:
- $a_{ij} \geq 0$ .
- $\sum_j a_{ij} = 1$ .

- Естественная задача: с какой вероятностью выпадет та или иная последовательность событий?
- Т.е. найти нужно для последовательности  $Q = q_{i_1} \dots q_{i_k}$

$$p(Q|\text{модель}) = p(q_{i_1})p(q_{i_2}|q_{i_1}) \dots p(q_{i_k}|q_{i_{k-1}}).$$

- Казалось бы, это тривиально.
- Что же сложного в реальных задачах?



- А сложно то, что никто нам не скажет, что модель должна быть именно такой.
- И, кроме того, мы обычно наблюдаем не  $x(t)$ , т.е. реальные состояния модели, а  $y(t)$ , т.е. некоторую функцию от них (данные).
- Пример: распознавание речи.

Спасибо за внимание!