Алгоритм EM и его применения

Сергей Николенко

Computer Science Club, Казань, 2014

Outline

Частный случай

- Алгоритм EM – это алгоритм вывода в вероятностных моделях, в которых есть скрытые переменные со сложными связями.
- Начнём с простейшего примера применения алгоритма EM:
  - 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:

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

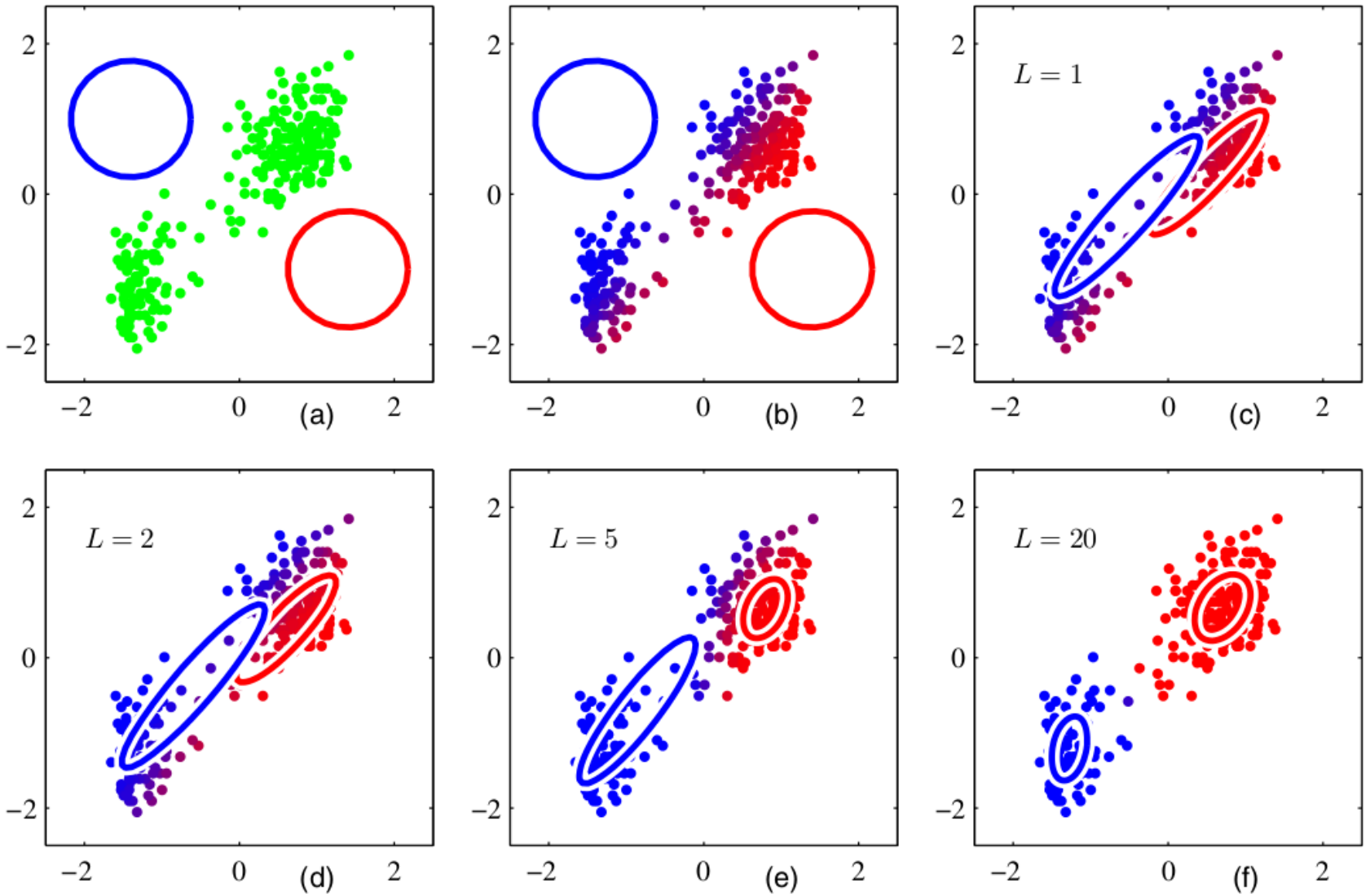
Мы подсчитываем эти ожидания, а потом подправляем гипотезу:

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

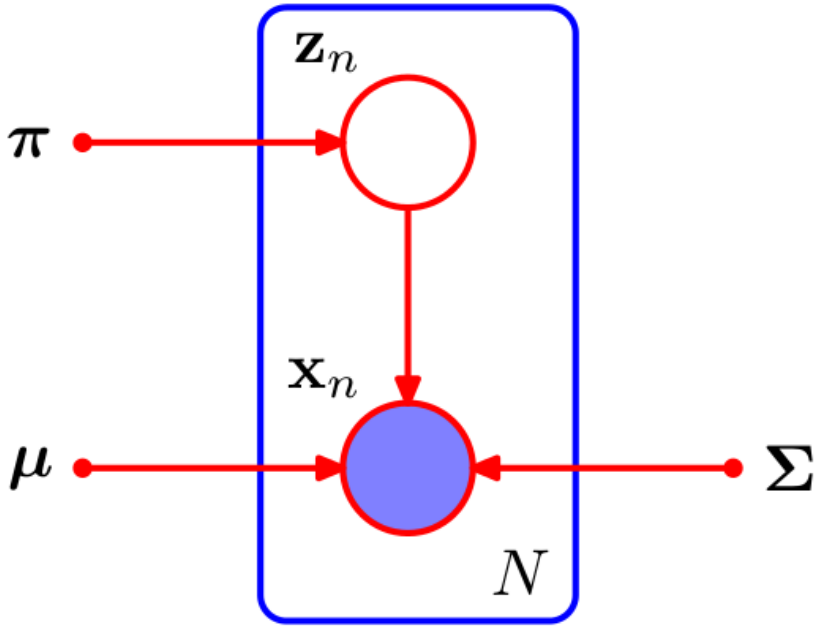
Алгоритм кластеризации

- Получился простой алгоритм гауссовской кластеризации:
  1. инициализировать центры кластеров $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_k)$;
  2. вычислить ожидания скрытых переменных $E(z_{ij}) = \frac{p(x=x_i|\mu=\mu_j)}{\sum_{j'} p(x=x_i|\mu=\mu_{j'})}$;
  3. пересчитать центры кластеров $\mu_j = \frac{1}{N} \sum_{i=1}^{N} E(z_{ij}) x_i$;
  4. повторять шаги 2-3 до сходимости.

EM в картинках [Bishop]



Графическая модель EM



Графическая модель такой кластеризации.

EM в общем случае

- 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.

EM в общем случае

- 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).

EM в общем случае

- 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:

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

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

EM в общем случае

- 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.

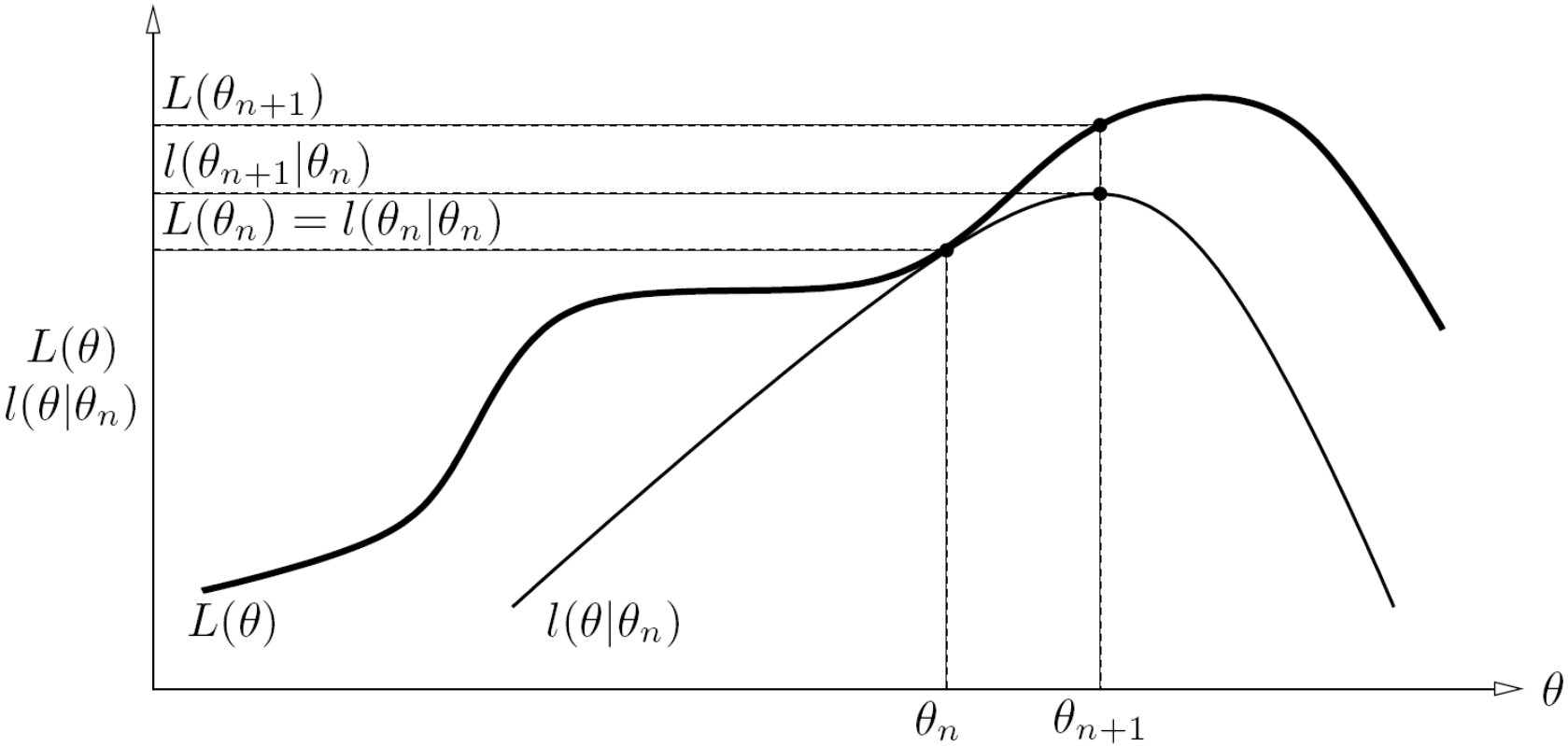
EM в общем случае

- 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.
- О том, почему это работает, сейчас не будем подробно – работает как minorization–maximization схема.

Обоснование алгоритма EM



Outline

Марковские цепи

- 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.

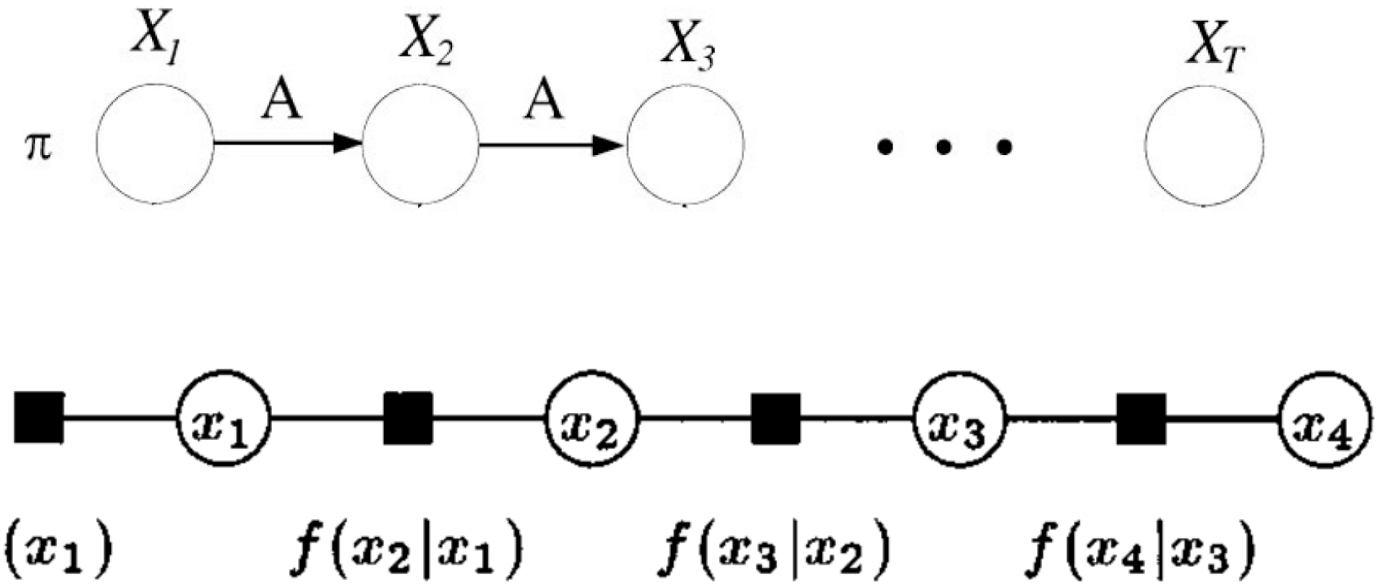
Дискретные марковские цепи

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

$$p(x(t) = x_j | x(t-1) = x_{j_{t-1}}, \ldots, x(1) = x_{j_1}) = \\ = p(x(t) = x_j | x(t-1) = x_{j_{t-1}}).$$

- 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 $A = (a_{ij})$.

Графическая модель марковской цепи



Прямая задача

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

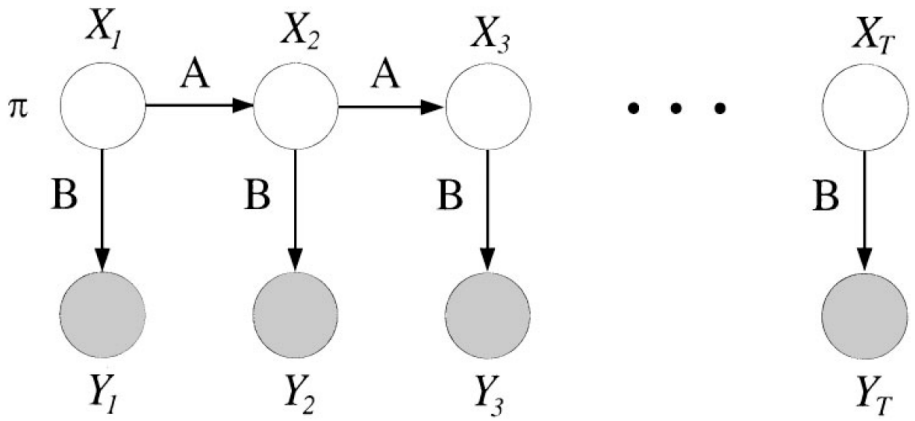
$$p(Q|\text{модель}) = p(q_{i_1})p(q_{i_2}|q_{i_1}) \dots p(q_{i_k}|q_{i_{k-1}}).$$

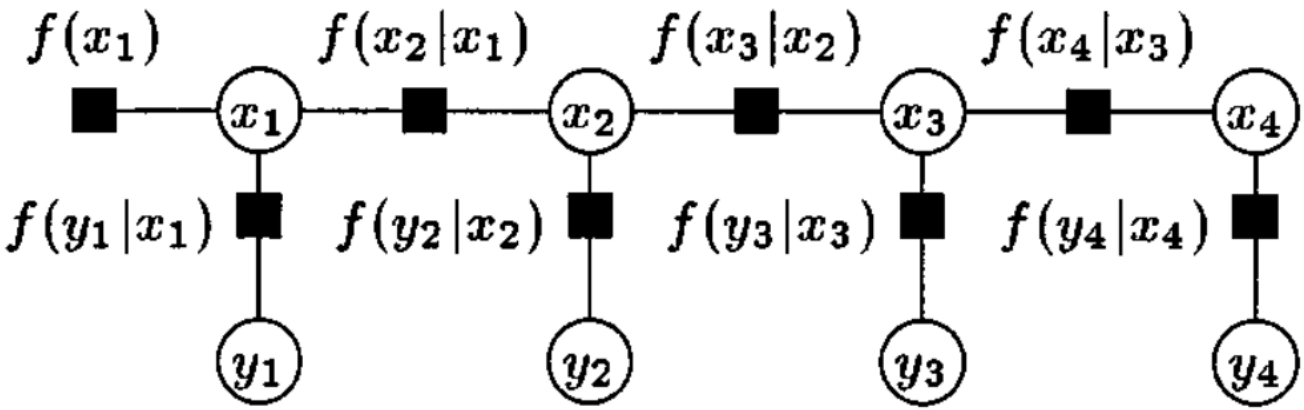
- Казалось бы, это тривиально.
- Что же сложного в реальных задачах?

Скрытые марковские модели

- 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.

Графическая модель скрытой марковской цепи





Задачи скрытых марковских моделей

- Первая: найти вероятность последовательности наблюдений в данной модели.
- Вторая: найти «оптимальную» последовательность состояний при условии данной модели и данной последовательности наблюдений.
- Третья: найти наиболее правдоподобную модель (параметры модели).

Состояния и наблюдаемые

- $X = \{x_1, \ldots, x_n\}$ — множество состояний.
- $V = \{v_1, \ldots, v_m\}$ — алфавит, из которого мы выбираем наблюдаемые y (множество значений y).
- q_t — состояние во время t, y_t — наблюдаемая во время t.

Распределения

- $a_{ij} = p(q_{t+1} = x_j | q_t = x_i)$ — вероятность перехода из i в j.
- $b_j(k) = p(v_k | x_j)$ — вероятность получить данные v_k в состоянии j.
- Начальное распределение $\pi = \{\pi_j\}$, $\pi_j = p(q_1 = x_j)$.
- Данные будем обозначать через $D = d_1 \dots d_T$ (последовательность наблюдаемых, d_i принимают значения из V).

Комментарий

- Проще говоря, вот как работает HMM (hidden Markov model).
- Выберем начальное состояние x_1 по распределению π.
- По t от 1 до T:
  - Выберем наблюдаемую d_t по распределению $p(v_k|x_j)$.
  - Выберем следующее состояние по распределению $p(q_{t+1} = x_j|q_t = x_i)$.
- Таким алгоритмом можно выбрать случайную последовательность наблюдаемых.

Задачи

- Теперь можно формализовать постановку задач.
- Первая задача: по данной модели $\lambda = (A, B, \pi)$ и последовательности D найти $p(D|\lambda)$. Фактически, это нужно для того, чтобы оценить, насколько хорошо модель подходит к данным.
- Вторая задача: по данной модели λ и последовательности D найти «оптимальную» последовательность состояний $Q = q_1 \dots q_T$. Как и раньше, будет два решения: «побитовое» и общее.
- Третья задача: оптимизировать параметры модели $\lambda = (A, B, \pi)$ так, чтобы максимизировать $p(D|\lambda)$ при данном D (найти модель максимального правдоподобия). Эта задача — главная, в ней и заключается обучение скрытых марковских моделей.

Постановка первой задачи

- Формально, первая задача выглядит так. Нужно найти

$$p(D|\lambda) = \sum_{Q} p(D|Q,\lambda)p(D|\lambda) =$$

$$= \sum_{q_1,\ldots,q_T} b_{q_1}(d_1)\ldots b_{q_T}(d_T)\pi_{q_1}a_{q_1q_2}\ldots a_{q_{T-1}q_T}.$$

- Ничего не напоминает?

Суть решения первой задачи

- Правильно, это такая же задача маргинализации, как мы решаем всё время.
- Мы воспользуемся де-факто алгоритмом передачи сообщений, просто в этом случае он просто записывается в виде динамического программирования.
- Будем последовательно вычислять промежуточные величины вида

$$\alpha_t(i) = p(d_1 \dots d_t, q_t = x_i|\lambda),$$

т.е. искомые вероятности, но ещё с учётом текущего состояния.

Решение первой задачи

- Инициализируем $\alpha_1(i) = \pi_i b_i(d_1)$.
- Шаг индукции:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{n} \alpha_t(i) a_{ij}\right] b_j(d_{t+1}).$$

- После того как дойдём до шага T, подсчитаем то, что нам нужно:

$$p(D|\lambda) = \sum_{i=1}^{n} \alpha_T(i).$$

- Фактически, это только прямой проход, обратный нам здесь не понадобился.
- Что вычислял бы обратный проход?

Обратный проход

- Он вычислял бы условные вероятности $\beta_t(i) = p(d_{t+1} \dots d_T | q_t = x_i, \lambda)$.
- Их можно вычислить, проинициализировав $\beta_T(i) = 1$, а затем по индукции:

$$\beta_t(i) = \sum_{j=1}^{n} a_{ij} b_j(d_{t+1}) \beta_{t+1}(j).$$

- Это нам пригодится чуть позже, при решении второй и третьей задачи.

Два варианта второй задачи

- Как мы уже упоминали, возможны два варианта.
- Первый: решать «побитово», отвечая на вопрос «какое наиболее вероятное состояние во время j?».
- Второй: решать задачу «какая наиболее вероятная последовательность состояний?».

Побитовое решение

- Рассмотрим вспомогательные переменные

$$\gamma_t(i) = p(q_t = x_i | D, \lambda).$$

- Наша задача – найти

$$q_t = \arg\max_{1 \le i \le n} \gamma_t(i), \quad 1 \le t \le T.$$

- Как это сделать?

Побитовое решение

- Выражаем через α и β:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{p(D|\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{n} \alpha_t(i)\beta_t(i)}.$$

- На знаменатель можно не обращать внимания — нам нужен $\arg\max$.

Решение относительно последовательности

- Чтобы ответить на вопрос о наиболее вероятной последовательности, мы будем использовать так называемый *алгоритм Витерби* (то есть, по сути, то же самое динамическое программирование).
- Наши вспомогательные переменные — это

$$\delta_t(i) = \max_{q_1,\ldots,q_{t-1}} p\left(q_1 q_2 \ldots q_t = x_i, d_1 d_2 \ldots d_t | \lambda\right).$$

Решение относительно последовательности

- Т.е. $\delta_t(i)$ — максимальная вероятность достичь состояния x_i на шаге t среди всех путей с заданными наблюдаемыми.
- По индукции:

$$\delta_{t+1}(j) = \left[\max_i \delta_t(i) a_{ij}\right] b_j(d_{t+1}).$$

- И надо ещё запоминать аргументы, а не только значения; для этого будет массив $\psi_t(j)$.

Решение относительно последовательности: алгоритм

- Проинициализируем $\delta_1(i) = \pi_i b_i(d_1)$, $\psi_1(i) = []$.
- Индукция:

$$\delta_t(j) = \max_{1 \le i \le n} [\delta_{t-1}(i) a_{ij}] \, b_j(d_t),$$

$$\psi_t(j) = \arg\max_{1 \le i \le n} [\delta_{t-1}(i) a_{ij}] \, .$$

- Когда дойдём до шага T, финальный шаг:

$$p^* = \max_{1 \le i \le n} \delta_T(i), \qquad q_T^* = \arg\max_{1 \le i \le n} \delta_T(i).$$

- И вычислим последовательность: $q_t^* = \psi_{t+1}(q_{t+1}^*)$.

Общая суть третьей задачи

- Аналитически найти глобальный максимум $p(D|\lambda)$ у нас никак не получится.
- Зато мы рассмотрим итеративную процедуру, которая приведёт к локальному максимуму.
- Это называется алгоритм Баума–Велха (Baum–Welch algorithm). Он является на самом деле частным случаем алгоритма EM.

Вспомогательные переменные

- Теперь нашими вспомогательными переменными будут вероятности того, что мы во время t в состоянии x_i, а во время $t+1$ — в состоянии x_j:

$$\xi_t(i,j) = p(q_t = x_i, q_{t+1} = x_j | D, \lambda).$$

- Если переписать через уже знакомые переменные:

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(d_{t+1}) \beta_{t+1}(j)}{p(D|\lambda)} = \frac{\alpha_t(i) a_{ij} b_j(d_{t+1}) \beta_{t+1}(j)}{\sum_i \sum_j \alpha_t(i) a_{ij} b_j(d_{t+1}) \beta_{t+1}(j)}.$$

- Отметим также, что $\gamma_t(i) = \sum_j \xi_t(i,j)$.

Идея

- $\sum_t \gamma_t(i)$ — это ожидаемое количество переходов из состояния x_i, а $\sum_t \xi_t(i,j)$ — из x_i в x_j.
- Теперь на шаге M мы будем переоценивать вероятности:

$$\bar{\pi}_i = \text{ожидаемая частота в } x_i \text{ на шаге } 1 = \gamma_1(i),$$

$$\bar{a}_{ij} = \frac{\text{к-во переходов из } x_i \text{ в } x_j}{\text{к-во переходов из } x_i} = \frac{\sum_t \xi_t(i,j)}{\sum_t \gamma_t(i)}.$$

$$\bar{b}_j(k) = \frac{\text{к-во появлений в } x_i \text{ и наблюдений } v_k}{\text{к-во появлений в } x_i} = \frac{\sum_{t:d_t=v_k} \gamma_t(i)}{\sum_t \gamma_t(i)}.$$

- EM-алгоритм приведёт к цели: начать с $\lambda = (A, B, \pi)$, подсчитать $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$, снова пересчитать параметры и т.д.

Thank you!

Спасибо за внимание!

Применительно к HMM

- Мы определим

$$p_1(Q) = \frac{p(Q, D|\lambda)}{p(D|\lambda)}, \quad p_2(Q) = \frac{p(Q, D|\lambda')}{p(D|\lambda')}.$$

- Тогда p_1 и p_2 — распределения, и расстояние Кульбака–Лейблера:

$$0 \leq D_{LK}(\lambda, \lambda') = \sum_Q \frac{p(Q, D|\lambda)}{p(D|\lambda)} \log \frac{p(Q, D|\lambda) p(D|\lambda')}{p(Q, D|\lambda') p(D|\lambda)} =$$

$$= \log \frac{p(D|\lambda')}{p(D|\lambda)} + \sum_Q \frac{p(Q, D|\lambda)}{p(D|\lambda)} \log \frac{p(Q, D|\lambda)}{p(Q, D|\lambda')}.$$

Вспомогательная функция

- Введём вспомогательную функцию

$$\boldsymbol{Q}(\lambda, \lambda') = \sum_{Q} p(Q|D, \lambda) \log p(Q|D, \lambda').$$

- Тогда из неравенства следует, что

$$\frac{\boldsymbol{Q}(\lambda, \lambda') - \boldsymbol{Q}(\lambda, \lambda)}{p(D|\lambda)} \leq \log \frac{p(D|\lambda')}{p(D|\lambda)}.$$

- Т.е., если $\boldsymbol{Q}(\lambda, \lambda') > \boldsymbol{Q}(\lambda, \lambda)$, то $p(D|\lambda') > p(D|\lambda)$.
- Т.е., если мы максимизируем $\boldsymbol{Q}(\lambda, \lambda')$ по λ', мы тем самым будем двигаться в нужную сторону.

Функция Q

- Нужно максимизировать $Q(\lambda, \lambda')$. Перепишем:

$$Q(\lambda, \lambda') = \sum_Q p(Q|D, \lambda) \log p(Q|D, \lambda') =$$

$$= \sum_Q p(Q|D, \lambda) \log \pi_{q_1} \prod_t a_{q_{t-1} q_t} b_{q_t}(d_t) =$$

$$= \sum_Q p(Q|D, \lambda) \log \pi_{q_1} + \sum_Q p(Q|D, \lambda) \sum_t \log a_{q_{t-1} q_t} b_{q_t}(d_t).$$

- Последнее выражение легко дифференцировать по a_{ij}, $b_i(k)$ и π_i, добавлять соответствующие множители Лагранжа и решать. Получится именно пересчёт по алгоритму Баума–Велха (проверьте!).