

# Скрытые марковские модели

Сергей Николенко

Machine Learning — CS Club, весна 2008

# Outline

- 1 Скрытые марковские модели: основное
  - Марковские цепи
  - Возникающие задачи
  - Решения задач
- 2 Специальные виды марковских моделей
  - Смеси выпуклых распределений
  - Продолжительность состояния
- 3 Разное
  - Авторегрессивные марковские модели
  - Критерии оптимизации HMM: ML, MMI, MDI
  - Сравнения, начальные значения и недостающие данные

## Марковские цепи

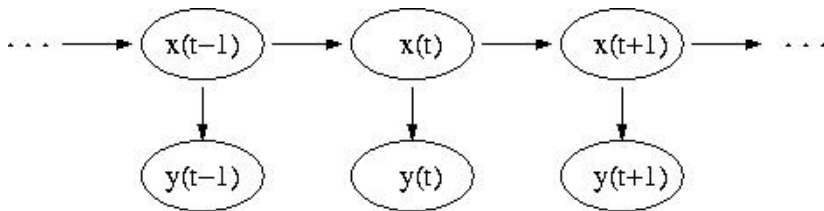
- Марковская цепь задаётся начальным распределением вероятностей  $p^0(x)$  и вероятностями перехода  $T(x'; x)$ .
- $T(x'; x)$  — это распределение следующего элемента цепи в зависимости от следующего; распределение на  $(t + 1)$ -м шаге равно

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

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

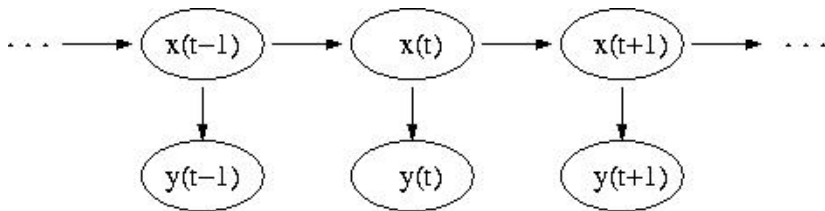
# Дискретные марковские цепи

- 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.

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

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

## Вероятности перехода

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

## 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).
- Давайте приведём пример.

## Скрытые марковские модели: пример

- Пусть кто-то бросает монетку и сообщает нам результаты — последовательность орлов и решек.
- Но мы не знаем, что он бросает монетку, мы только знаем, что есть вот такая последовательность битов.
- Если мы предположим, что он бросает одну монетку, модель будет одна: два состояния, вероятности перехода между ними  $p$  и  $1 - p$ , вероятности остаться  $1 - p$  и  $p$ .

## Скрытые марковские модели: пример

- Но мы же можем подумать, что у него две монетки! :)
- Тогда состояния по-прежнему два, но параметров больше.
- А если три монетки?..
- В общем, задачи уже ясны.

## 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

- $X = \{x_1, \dots, x_n\}$  — множество состояний.
- $V = \{v_1, \dots, v_m\}$  — 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  $\pi$ .
- 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  $\lambda$  и последовательности  $D$  найти «оптимальную» последовательность состояний  $Q = q_1 \dots q_T$ . Как и раньше, будет два решения: «битовое» 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

$$\begin{aligned} p(D|\lambda) &= \sum_Q p(D|Q, \lambda) p(Q|\lambda) = \\ &= \sum_{q_1, \dots, q_T} b_{q_1}(d_1) \dots b_{q_T}(d_T) \pi_{q_1} a_{q_1 q_2} \dots a_{q_{T-1} q_T}. \end{aligned}$$

- Ничего не напоминает?

## Суть решения первой задачи

- Правильно, это такая же задача маргинализации, как мы решаем всё время.
- Мы воспользуемся так называемой forward–backward procedure, по сути — вычислением на решётке, как в декодировании.
- 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

$$q_t = \operatorname{argmax}_{1 \leq i \leq n} \gamma_t(i), \quad 1 \leq t \leq T.$$

- Как это сделать?

## Побитовое решение

- Выражаем через  $\alpha$  и  $\beta$ :

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

- На знаменатель можно не обращать внимания — нам нужен  $\operatorname{argmax}$ .

## Решение относительно последовательности

- Чтобы ответить на вопрос о наиболее вероятной последовательности, мы будем использовать уже знакомый алгоритм Витерби.
- То есть, по сути, то же самое динамическое программирование.
- Наши вспомогательные переменные — это

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



## 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  $\delta_1(i) = \pi_i b_i(d_1)$ ,  $\psi_1(i) = \square$ .
- Индукция:

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

$$\psi_t(j) = \operatorname{argmax}_{1 \leq i \leq n} [\delta_{t-1}(i) a_{ij}].$$

- Когда дойдём до шага  $T$ , финальный шаг:

$$p^* = \max_{1 \leq i \leq n} \delta_T(i), \quad q_T^* = \operatorname{argmax}_{1 \leq i \leq n} \delta_T(i).$$

- 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 (по сути — градиентный подъём), 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:

$\bar{\pi}_i =$  ожидаемая частота в  $x_i$  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.

## Расстояние Кульбака–Лейблера

- Kullback–Leibler distance (divergence) — это информационно-теоретическая мера того, насколько далеки распределения друг от друга.

$$D_{KL}(p_1, p_2) = \sum_x p_1(x) \log \frac{p_1(x)}{p_2(x)}.$$

- Известно, что это расстояние всегда неотрицательно, равно нулю iff  $p_1 \equiv p_2$ .

## Применительно к НММ

- 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 Kullback–Leibler:

$$\begin{aligned} 0 \leq D_{LK}(\lambda, \lambda') &= \sum_Q \frac{p(Q, D|\lambda)}{p(D|\lambda)} \log \frac{p(Q, D|\lambda)p(D|\lambda')}{p(Q, D|\lambda')p(D|\lambda)} = \\ &= \log \frac{p(D|\lambda')}{p(D|\lambda)} + \sum_Q \frac{p(Q, D|\lambda)}{p(D|\lambda)} \log \frac{p(Q, D|\lambda)}{p(Q, D|\lambda')}. \end{aligned}$$

# Вспомогательная функция

- Введём вспомогательную функцию

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

- Тогда из неравенства следует, что

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

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



Функция  $Q$ 

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

$$\begin{aligned} Q(\lambda, \lambda') &= \sum_Q p(Q|D, \lambda) \log p(Q|D, \lambda') = \\ &= \sum_Q p(Q|D, \lambda) \log \pi_{q_1} \prod_t a_{q_{t-1}q_t} b_{q_t}(d_t) = \\ &= \sum_Q p(Q|D, \lambda) \log \pi_{q_1} + \sum_Q p(Q|D, \lambda) \sum_t \log a_{q_{t-1}q_t} b_{q_t}(d_t). \end{aligned}$$

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

# Outline

- 1 Скрытые марковские модели: основное
  - Марковские цепи
  - Возникающие задачи
  - Решения задач
- 2 Специальные виды марковских моделей
  - Смеси выпуклых распределений
  - Продолжительность состояния
- 3 Разное
  - Авторегрессивные марковские модели
  - Критерии оптимизации HMM: ML, MMI, MDI
  - Сравнения, начальные значения и недостающие данные

## Специальные виды моделей

- НММ эргодична, если  $\forall i, j \ a_{ij} > 0$ .
- НММ ациклична (left-right model, Bakis model), если  $\forall j < i \ a_{ij} = 0$  (матрица треугольная) и  $\pi_1 = 1$  (начинаем всегда в состоянии 1).
- В ациклических сетях ещё часто бывает ограничение на прыжок (constrained jump) размером  $\Delta$ :  $\forall j > i + \Delta \ a_{ij} = 0$  (матрица мультидиагональная).

## Непрерывные плотности наблюдаемых

- У нас были дискретные наблюдаемые с вероятностями  $B = (b_j(k))$ .
- Но в реальной жизни всё сложнее: зачастую мы наблюдаем непрерывные сигналы, а не дискретные величины, и дискретизовать их или плохо, или неудобно.
- При этом саму цепь можно оставить дискретной, т.е. перейти к непрерывным  $b_j(D)$ .

## Специальный вид плотности

- Не для всех плотностей найдены алгоритмы пересчёта (обобщения алгоритма Баума–Велха).
- Наиболее общий результат верен, когда  $b_j(D)$  можно представить в виде

$$b_j(D) = \sum_{m=1}^M c_{jm} \mathcal{P}(D, \mu_{jm}, \sigma_{jm}),$$

где  $c_{jm}$  — коэффициенты смеси ( $\sum_m c_{jm} = 1$ ), а  $\mathcal{P}$  — выпуклое распределение со средним  $\mu$  и вариацией  $\sigma$  (гауссиан подойдёт).

- К счастью, такой конструкцией можно приблизить любое непрерывное распределение, поэтому это можно широко применять.

# Вспомогательные переменные

- $\gamma_t(j, m)$  — вероятность быть в состоянии  $j$  во время  $t$ , причём за  $D$  отвечает  $m$ -й компонент смеси.
- Формально говоря,

$$\gamma_t(j, m) = \left[ \frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^N \alpha_t(j)\beta_t(j)} \right] \left[ \frac{c_{jm}\mathcal{P}(d_t, \mu_{jm}, \sigma_{jm})}{\sum_{m=1}^M c_{jm}\mathcal{P}(d_t, \mu_{jm}, \sigma_{jm})} \right].$$

- Если  $M = 1$ , то это уже известные нам  $\gamma_t(j)$ .

## Алгоритм для этого случая

- Нужно научиться пересчитывать  $b_j(D)$ , т.е. пересчитывать  $c_{jm}$ ,  $\mu_{jm}$  и  $\sigma_{jm}$ .
- Это делается так:

$$\bar{c}_{jm} = \frac{\sum_{t=1}^T \gamma_t(j, m)}{\sum_{t=1}^T \sum_{m=1}^M \gamma_t(j, m)},$$

$$\bar{\mu}_{jm} = \frac{\sum_{t=1}^T \gamma_t(j, m) \cdot d_t}{\sum_{t=1}^T \gamma_t(j, m)},$$

$$\bar{\sigma}_{jm} = \frac{\sum_{t=1}^T \gamma_t(j, m) \cdot (d_t - \mu_{jm})(d_t - \mu_{jm})^t}{\sum_{t=1}^T \gamma_t(j, m)}.$$

## Проблема

- Как моделировать продолжительность нахождения в том или ином состоянии?
- В дискретном случае вероятность пробыть в состоянии  $i$   $d$  шагов:

$$p_i(d) = a_{ii}^{d-1}(1 - a_{ii}).$$

- Однако для большинства физических сигналов такое экспоненциальное распределение не соответствует действительности. Мы бы хотели явно задавать плотность пребывания в данном состоянии.
- Т.е. вместо коэффициентов перехода в себя  $a_{ii}$  — явное задание распределения  $p_i(d)$ .



## Вспомогательные переменные

- Введём переменные

$$\alpha_t(i) = p(d_1 \dots d_t, x_i \text{ заканчивается во время } t|\lambda).$$

- Всего за первые  $t$  шагов посещено  $r$  состояний  $q_1 \dots q_r$ , и мы там оставались  $d_1, \dots, d_r$ . Т.е. ограничения:

$$q_r = x_i, \quad \sum_{s=1}^r d_s = t.$$

Вычисление  $\alpha_t(i)$ 

- Тогда получается

$$\begin{aligned}\alpha_t(i) = & \sum_q \sum_d \pi_{q_1} p_{q_1}(d_1) p(d_1 d_2 \dots d_{d_1} | q_1) \\ & a_{q_1 q_2} p_{q_2}(d_2) p(d_{d_1+1} \dots d_{d_1+d_2} | q_2) \dots \\ & \dots a_{q_{r-1} q_r} p_{q_r}(d_r) p(d_{d_1+\dots+d_{r-1}+1} \dots d_t | q_r).\end{aligned}$$

Вычисление  $\alpha_t(i)$ 

- По индукции

$$\alpha_t(j) = \sum_{i=1}^n \sum_{d=1}^D \alpha_{t-d}(i) a_{ij} p_j(d) \prod_{s=t-d+1}^t b_j(d_s),$$

где  $D$  — максимальная остановка в любом из состояний.

- Тогда, как и раньше,

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

## Вспомогательные переменные

- Для пересчёта потребуются ещё три переменные:

$$\alpha_t^*(i) = p(d_1 \dots d_t, x_i \text{ начинается во время } t + 1 | \lambda),$$

$$\beta_t(i) = p(d_{t+1} \dots d_T | x_i \text{ заканчивается во время } t, \lambda),$$

$$\beta_t^*(i) = p(d_{t+1} \dots d_T | x_i \text{ начинается во время } t + 1, \lambda).$$

# Вспомогательные переменные

- Соотношения между ними:

$$\alpha_t^*(j) = \sum_{i=1}^n \alpha_t(i) a_{ij},$$

$$\alpha_t(i) = \sum_{d=1}^D \alpha_{t-d}^*(i) p_i(d) \prod_{s=t-d+1}^t b_i(d_s),$$

$$\beta_t(i) = \sum_{j=1}^n a_{ij} \beta_t^*(j),$$

$$\beta_t^*(i) = \sum_{d=1}^D \beta_{t+d}(i) p_i(d) \prod_{s=t+1}^{t+d} b_i(d_s).$$

## Формулы пересчёта

- Приведём формулы пересчёта.
- $\pi_i$  — просто вероятность того, что  $x_i$  был первым состоянием:

$$\hat{\pi}_i = \frac{\pi_i \beta_0^*(i)}{p(d|\lambda)}.$$

- $a_{ij}$  — та же формула, что обычно, только вместе с  $\alpha$  есть ещё и  $\beta$ , которая говорит, что новое состояние начинается на следующем шаге:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^T \alpha_t(i) a_{ij} \beta_t^*(j)}{\sum_{k=1}^n \sum_{t=1}^T \alpha_t(i) a_{ik} \beta_t^*(k)}.$$

## Формулы пересчёта

- $b_i(k)$  — отношение ожидания количества событий  $d_t = v_k$  в состоянии  $x_i$  к ожиданию количества любого  $v_j$  в состоянии  $x_i$ :

$$\hat{b}_i(k) = \frac{\sum_{t=1, d_t=v_k}^T (\sum_{\tau < t} \alpha_{\tau}^*(i) \beta_{\tau}^*(i) - \sum_{\tau < t} \alpha_{\tau}(i) \beta_{\tau}(i))}{\sum_{k=1}^m \sum_{t=1, d_t=v_k}^T (\sum_{\tau < t} \alpha_{\tau}^*(i) \beta_{\tau}^*(i) - \sum_{\tau < t} \alpha_{\tau}(i) \beta_{\tau}(i))}.$$

- $p_i(d)$  — отношение ожидания количества раз, которые  $x_i$  случилось с продолжительностью  $d$ , к количеству раз, которые  $x_i$  вообще случилось:

$$\hat{p}_i(d) = \frac{\sum_{t=1}^T \alpha_t^*(i) p_i(d) \beta_{t+d}(i) \prod_{s=t+1}^{t+d} b_i(d_s)}{\sum_{d=1}^D \sum_{t=1}^T \alpha_t^*(i) p_i(d) \beta_{t+d}(i) \prod_{s=t+1}^{t+d} b_i(d_s)}.$$

## За и против

- Такой подход очень полезен, когда  $p_i(d)$  далеко от экспоненциального.
- Однако он сильно увеличивает вычислительную сложность (в  $D^2$  раз).
- И, кроме того, становится гораздо больше параметров, т.е. нужно, вообще говоря, больше данных, чтобы эти параметры надёжно оценить.



# Параметрическая продолжительность состояния

- Чтобы уменьшить количество параметров, можно иногда считать, что  $p_i(d)$  — классическое распределение с не слишком большим количеством параметров.
- Например,  $p_i(d)$  может быть равномерным, или нормальным ( $p_i(d) = \mathcal{N}(d, \mu_i, \sigma_i^2)$ ), или гамма-распределением:

$$p_i(d) = \frac{\eta_i^{\nu_i} d^{\nu_i-1} e^{-\eta_i d}}{\Gamma(\nu_i)}.$$

# Outline

- 1 Скрытые марковские модели: основное
  - Марковские цепи
  - Возникающие задачи
  - Решения задач
- 2 Специальные виды марковских моделей
  - Смеси выпуклых распределений
  - Продолжительность состояния
- 3 Разное
  - Авторегрессивные марковские модели
  - Критерии оптимизации HMM: ML, MMI, MDI
  - Сравнения, начальные значения и недостающие данные

## Общая идея

- Речь идёт о моделях, в которых следующие компоненты вектора наблюдаемых зависят от предыдущих с некоторыми коэффициентами.
- Т.е.  $d = (d_1, \dots, d_K)$  — вектор наблюдаемых, и модель такая:

$$d_k = - \sum_{i=1}^p a_i d_{k-i} + e_k,$$

где  $e_k$  — гауссианы с нулевым средним и вариацией  $\sigma^2$ ,  
 $a_i$  — коэффициенты авторегрессии.

# Распределение

- Можно показать, что плотность  $d$  для больших  $k$  стабилизируется на

$$p(d) = (2\pi\sigma^2)^{-K/2} e^{-\frac{1}{2\sigma^2} \delta(d,a)},$$

где (положив  $a_0 = 1$ )

$$\delta(d, a) = r_a(0)r(0) + 2 \sum_{i=1}^p r_a(i)r(i),$$

$$r_a(i) = \sum_{j=0}^{p-i} a_j a_{j+i}, \quad r(i) = \sum_{j=0}^{K-i-1} d_j d_{j+i}.$$

## Комментарий к распределению

- На самом деле  $r(i)$  — автокорреляция выборки, а  $r_a(i)$  — автокорреляция коэффициентов авторегрессии.
- Автокорреляция — это корреляция процесса с самим собой в предыдущие периоды времени; используется в статистике и обработке сигналов для поиска паттернов в данных.
- По определению, для процесса  $X_t$   $R(t, s) = \frac{E[(X_t - \mu)(X_s - \mu)]}{\sigma^2}$ , а если процесс стационарный (не зависит от конкретного времени), то

$$R(k) = \frac{E[(X_t - \mu)(X_{t+k} - \mu)]}{\sigma^2}.$$

- В обработке сигналов (а мы в ней) часто используют автокорреляцию без нормализации, т.е. для дискретного сигнала  $R(j) = \sum_k x_j x_{j-k}$ .

# Скрытая марковская модель

- Как применить авторегрессивную модель?
- Очень просто: сначала нормализуем вектор наблюдений:

$$\hat{d} = \frac{d}{\sqrt{K\sigma^2}}, \quad p(\hat{d}) = \left(\frac{2\pi}{K}\right)^{-K/2} e^{-\frac{K}{2}\delta(\hat{d},a)}.$$

- А затем рассмотрим смесь

$$b_j(D) = \sum_{m=1}^M c_{jm} b_{jm}(D),$$

где  $b_{jm}$  задаётся вектором авторегрессии  $a_{jm}$  (т.е. его коэффициентами  $r_{a_{jm}}$ ).

# Алгоритм пересчёта

- Надо научиться пересчитывать  $r_{ajm}$ . Для этого схема такая: сначала научимся пересчитывать  $r_{jm}$ , потом из них получим  $a_{jm}$ , а потом по ним пересчитаем  $r_{ajm}$ .



$$\bar{r}_{jm} = \frac{\sum_{t=1}^T \gamma_t(j, m) \cdot r_t}{\sum_{t=1}^T \gamma_t(j, m)},$$

где

$$\gamma_t(j, m) = \left[ \frac{\alpha_t(j) \beta_t(j)}{\sum_{j=1}^N \alpha_t(j) \beta_t(j)} \right] \left[ \frac{c_{jm} b_{jm}(d_t)}{\sum_{m=1}^M c_{jm} b_{jm}(d_t)} \right].$$

## Постановка задачи

- Мы раньше предполагали, что наша марковская модель хорошо описывает процесс.
- На самом деле это не всегда так.
- Давайте попробуем решить более практическую задачу: есть несколько сигналов, и мы должны их описать марковскими моделями так, чтобы как можно точнее отделять их друг от друга.



# ML

- Стандартная идея в том, что мы хотим научиться сравнивать модели, т.е. для разных моделей  $\lambda_\nu$ ,  $\nu = 1..V$ , научиться оценивать  $p(d^\nu | \lambda_\nu)$ , а потом выбрать из них максимум.
- Это так называемый ML criterion (от maximum likelihood).

## MMI

- Альтернатива — метод максимальной взаимной информации (maximum mutual information, MMI).
- В простейшей ситуации этот метод применяется, чтобы выяснить, какие параметры больше всего влияют на различение заданных классов.
- Если есть данные  $x = (x_1, \dots, x_n)$  и набор классов  $G$ , по которым их классифицируют, то нужно подсчитать информацию между  $x_i$  и классом  $C$ :

$$I(x_i, C) = H(C) - H(C|x_i),$$

т.е. максимизировать

$$H(C|x_i) = - \sum_{c \in G} \sum_{v_j} p(c, x_i = v_j) \log p(c|x_i = v_j).$$

## MMI

- Для марковских моделей это выражается как максимизация среднего расстояния между данными  $d$  и полным набором моделей  $\lambda = (\lambda_1, \dots, \lambda_V)$ :

$$l_v = \max_{\lambda} \left[ \log p(d|\lambda_v) - \log \sum_w p(d^v|\lambda_w) \right],$$

т.е. мы отделяем правильную модель  $\mu$  от других моделей на последовательности  $d$ .

- А теперь можно просуммировать по всем моделям и таким образом надеяться, что получится самый «разделённый» набор:

$$l = \max_{\lambda} \sum_{v=1}^V \left[ \log p(d^v|\lambda_v) - \log \sum_w p(d^v|\lambda_w) \right],$$

## MDI

- Третий подход: предположим, что сигнал не обязательно марковский, но у него есть некоторые ограничения (на корреляцию, например).
- Надо найти такие параметры, которые минимизируют разделяющую информацию (discrimination information, DI) между набором распределений  $Q$ , удовлетворяющих ограничениям, и набором марковских распределений  $p_\lambda$ :

$$D(Q||p_\lambda) = \int q(y) \ln \frac{q(y)}{p(y)} dy.$$

- Т.е. мы предполагаем, что сигнал из  $Q$ , и ищем такую  $\lambda$ , чтобы расстояние до соответствующего элемента  $q$  было минимальным.
- Тоже по сути модифицированный алгоритм Баума–Велха.

## Как сравнивать HMM?

- Вспомним пример с монеткой и рассмотрим две модели:

$$\lambda_1 = (A_1, B_1, \pi_1), \lambda_2 = (A_2, B_2, \pi_2):$$

$$A_1 = \begin{pmatrix} p & 1-p \\ 1-p & p \end{pmatrix}, B_1 = \begin{pmatrix} q & 1-q \\ 1-q & q \end{pmatrix}, \pi_1 = (1/2 \ 1/2),$$

$$A_2 = \begin{pmatrix} r & 1-r \\ 1-r & r \end{pmatrix}, B_2 = \begin{pmatrix} s & 1-s \\ 1-s & s \end{pmatrix}, \pi_2 = (1/2 \ 1/2),$$

- Чтобы модели производили в среднем одинаковые последовательности наблюдений, нужно, чтобы  $E[d_t = v_k | \lambda_1] = E[d_t = v_k | \lambda_2]$ , т.е.

$$pq + (1-p)(1-q) = rs + (1-r)(1-s).$$

- Модели с  $p = 0.6$ ,  $q = 0.7$  и  $r = 0.2$ ,  $s = 13/30$  дадут одинаковые результаты.

## Как сравнивать HMM?

- Нужна метрика. Определим расстояние между моделями

$$D(\lambda_1, \lambda_2) = \frac{1}{T} (\log p(d^{(2)}|\lambda_1) - \log p(d^{(2)}|\lambda_2)),$$

где  $d^{(2)}$  порождено моделью  $\lambda_2$ , т.е. мы проверяем, насколько хорошо  $\lambda_1$  соответствует наблюдениям по модели  $\lambda_2$  по сравнению с самой  $\lambda_2$ .

- Это, конечно, несимметричное расстояние, поэтому обычно берут

$$D_s(\lambda_1, \lambda_2) = \frac{D(\lambda_1, \lambda_2) + D(\lambda_2, \lambda_1)}{2}.$$

## Начальные значения параметров

- Алгоритм Баума–Велха, по сути своей градиентный, конечно, даёт только локальный максимум.
- Значит, важно, как выбирать начальные значения параметров.
- Для  $\pi$  и  $A$  на самом деле особой проблемы нет: обычно хорошо работают или случайные, или равномерные начальные значения.
- Для  $B$  хорошие начальные оценки могут быть весьма полезны в дискретном случае и практически необходимы в непрерывном.
- Как их получить?

## Начальные значения параметров

- В непрерывном случае у нас распределение было смесью других распределений:

$$b_j(D) = \sum_{m=1}^M c_{jm} \mathcal{P}(D, \mu_{jm}, \sigma_{jm}).$$

- Как бы нам оценить начальные параметры  $\mu_{jm}$ ,  $\sigma_{jm}$ , имея данные всех наблюдений?
- Можно применить кластеризацию. Мы просто кластеризуем данные  $D$  (по каждому времени  $j$  отдельно) и получим неплохие начальные значения.



# Интерполяция


- Данных часто не хватает. Параметров бывает слишком много.
- Один из путей — интерполяция. Выбрать вместе с  $\lambda$  модель поменьше  $\lambda'$ , для которой данных достаточно, и считать интерполированную модель

$$\hat{\lambda} = \epsilon\lambda + (1 - \epsilon)\lambda'.$$

# Интерполяция

- Главное — правильно оценить  $\epsilon$  (это функция количества имеющихся тестовых данных).
- Есть специальные подходы, при которых данные делятся на  $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$ , потом  $\mathcal{D}_1$  используется для тренировки, а  $\mathcal{D}_2$  — для оценки  $\epsilon$ . Разбиение можно со временем двигать.
- Другой путь — добавлять специальные ограничения (например,  $b_j(k) \geq \epsilon$ ).

## Спасибо за внимание!

- Lecture notes и слайды будут появляться на моей homepage:  
<http://logic.pdmi.ras.ru/~sergey/index.php?page=teaching>
- Присылайте любые замечания, решения упражнений, новые численные примеры и прочее по адресам:  
[sergey@logic.pdmi.ras.ru](mailto:sergey@logic.pdmi.ras.ru), [snikolenko@gmail.com](mailto:snikolenko@gmail.com)
- Заходите в ЖЖ  [smartnik](#).