

# Скрытые марковские модели

---

Сергей Николенко

НИУ ВШЭ — Санкт-Петербург

11 апреля 2020 г.

---

*Random facts:*

- 11 апреля 1868 г. Токугава Ёсинобу сдал замок Эдо имперским войскам, чем завершил славную историю сёгуната Токугава
- 11 апреля 1961 г. Боб Дилан начал исполнять свой первый контракт, две недели выступлений в Gerdes Folk City на 4th Street
- 11 апреля 1967 г. в лондонском Национальном театре прошла премьера первой пьесы Тома Стоппарда «Розенкранц и Гильденстерн мертвы»
- 11 апреля 1976 г. Стив Возняк построил компьютер Apple I; сначала он хотел опубликовать дизайн компьютера под открытой лицензией, но у его друга Стива Джобса была идея получше
- 11 апреля 1982 г. в «Комсомольской правде» вышла статья «Рагу из синей птицы»

# Скрытые марковские модели: основное

---

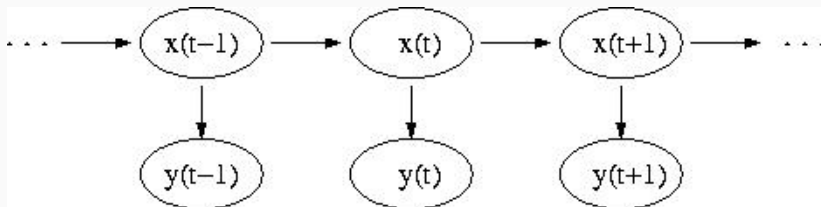
- Марковская цепь задаётся начальным распределением вероятностей  $p^0(x)$  и вероятностями перехода  $T(x'; x)$ .
- $T(x'; x)$  — это распределение следующего элемента цепи в зависимости от следующего; распределение на  $(t + 1)$ -м шаге равно

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

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

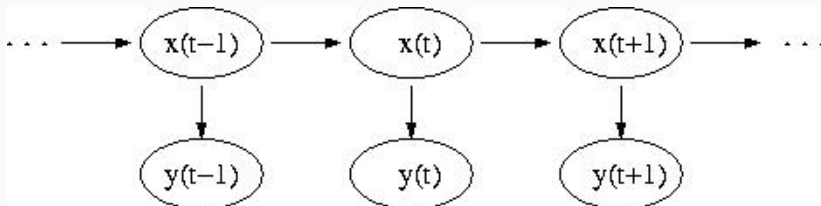
# Дискретные марковские цепи

- 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.

# 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).

- $X = \{x_1, \dots, x_n\}$  — 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  $t$ .

- $a_{ij} = p(q_{t+1} = x_j | q_t = x_i)$  — 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  $V$ ).

- Проще говоря, вот как работает НММ (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: «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

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

- 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 — динамическим программированием на решётке.
- 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?

- 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?».

- Рассмотрим вспомогательные переменные

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

- Наша задача – найти

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

- Как это сделать?

- 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  $\arg \max$ .

- 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

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

- 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:

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

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

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

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

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

- $\sum_t \gamma_t(i)$  — 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.

- 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  $p_1 \equiv p_2$ .

- 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.

- 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!).

# Специальные виды марковских моделей

---



- 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.

- $\gamma_t(j, m)$  — 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:

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

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

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

- 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  $p_i(d)$ .

- 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:

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

- Тогда получается

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

- По индукции

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

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

- Тогда, как и раньше,

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



- Для пересчёта потребуются ещё три переменные:

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

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

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

- 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  $x_i$ :

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

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

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

- 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.

- 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:

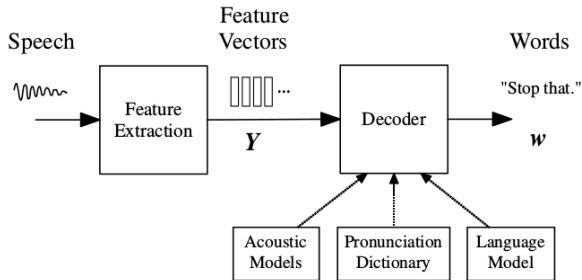
$$p_i(d) = \frac{\eta_i^{\nu_i} d^{\nu_i-1} e^{-\eta_i d}}{\Gamma(\nu_i)}.$$

# НММ для распознавания речи

---

# Общая структура

- НММ – классический подход к распознаванию речи.
- Сейчас, правда, они уже в основном заменены глубокими нейронными сетями; но всё равно полезно проследить, как их можно применить.





- Признаки как-то выделились, а потом слово  $w$  делится на фонемы  $\mathbf{q} = q_1 \dots q_{|w|}$ , и правдоподобие наблюдаемых признаков

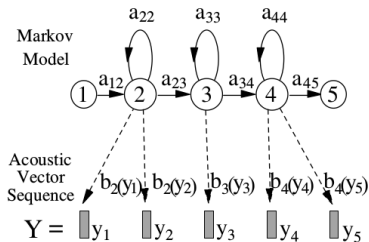
$$p(\mathbf{y} | \mathbf{w}) = \sum_{\mathbf{q}} p(\mathbf{y} | \mathbf{q}) p(\mathbf{q} | \mathbf{w}),$$

сумма по возможным произношениям (маленькая сумма), а  $\mathbf{w}$  – это все слова  $w_1 \dots w_L$ :

$$p(\mathbf{q} | \mathbf{w}) = \prod_{l=1}^L p(\mathbf{q}^{(w_l)} | w_l).$$

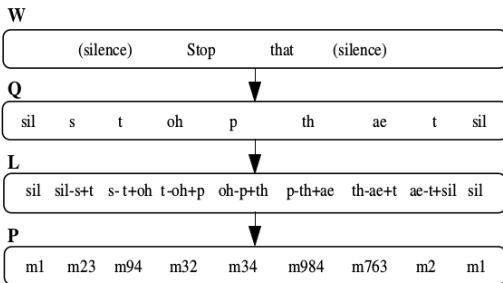
# Акустическая модель

- Каждая фонема – это HMM с непрерывными наблюдаемыми  $b_i(\mathbf{y}) = \mathcal{N}(\mathbf{y}; \mu^{(i)}, \Sigma^{(i)})$ .
- На этом месте уже можно обучать просто всё сразу.



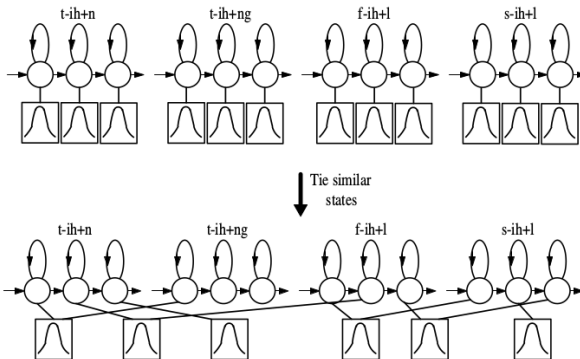
# Акустическая модель

- Однако фонемы очень по-разному звучат в зависимости от контекста.
- Можно перейти к трифонам.



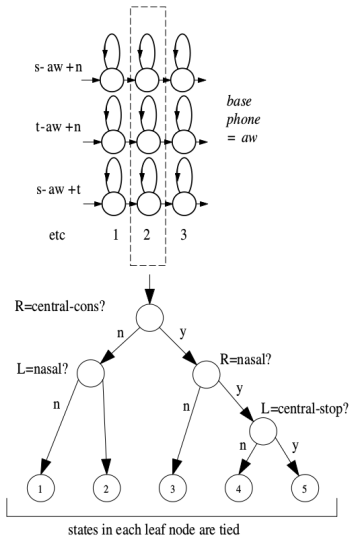
# Акустическая модель

- Но их будет целых  $N^3$ , и лучше объединить похожие и связать их параметры:



# Акустическая модель

- Это можно сделать просто силой мысли:



- Но это только начало. Ещё нужна *языковая модель* – мы о многом просто догадываемся.
- То есть нужно априорное распределение

$$p(\mathbf{w}) = \prod_{l=1}^L p(w_l \mid w_1 \dots w_{l-1}).$$

- Классический подход –  $n$ -граммы для  $n = 2..4$ :

$$p(\mathbf{w}) = \prod_{l=1}^L p(w_l \mid w_{l-1} \dots w_{l-n+1}).$$

- Качество языковых моделей сравнивают в терминах их *перплексии* (perplexity)

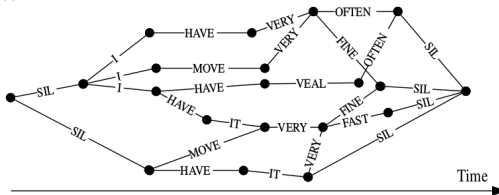
$$H = - \lim_{L \rightarrow \infty} \frac{1}{L} \log p(w_1, \dots, w_L).$$

- О современных языковых моделях мы обязательно поговорим потом...
- А пока декодер идёт и алгоритмом Витерби всё решает.
- Но что именно решает?

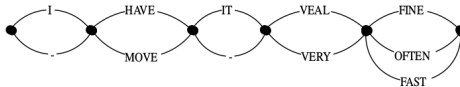
# Результаты

- Возможности удобно представлять как *решётку слов* (word lattice) или *confusion network*.

(a) Word Lattice



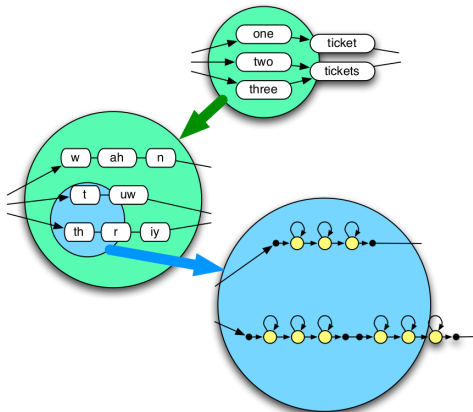
(b) Confusion Network





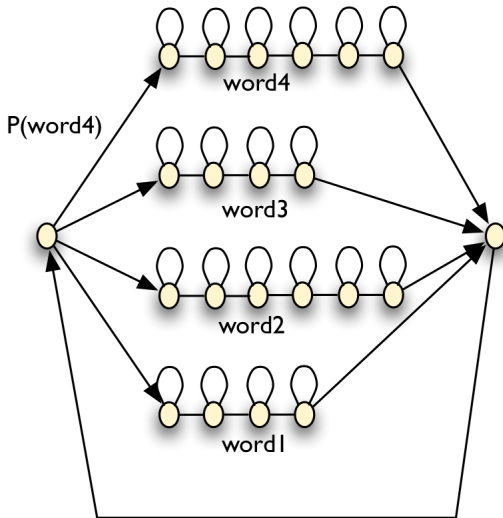
# Результаты

- Сеть слов превращается в сеть фонем, потом в сеть состояний HMM.



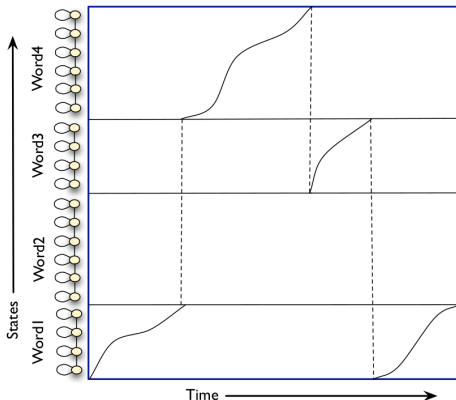
# Результаты

- И можно распознавать связанные друг с другом слова тоже алгоритмом Витерби.



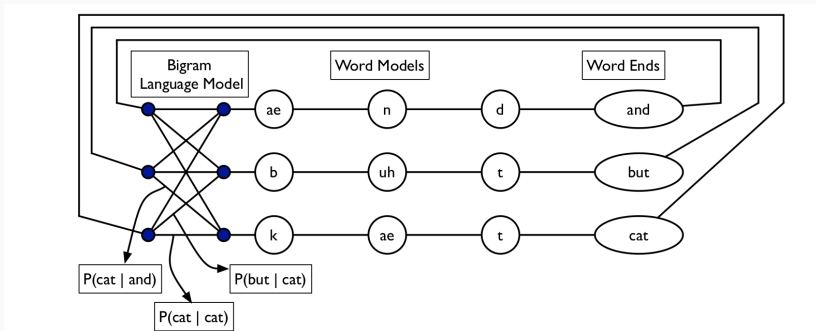
# Результаты

- Всё это накладывается на собственно аудиозапись, получается последовательность во времени.



# Результаты

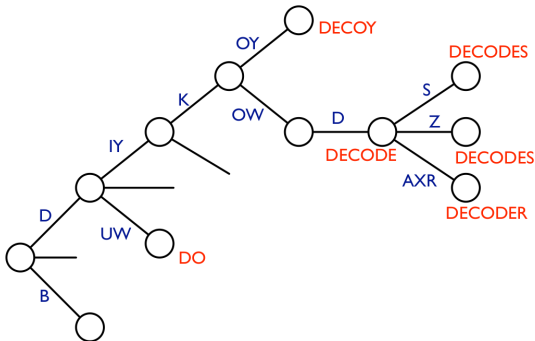
- А языковая модель – это просто дополнительные множители (слагаемые в  $\log$ ), априорные вероятности.



- Декодирование по всей сети всего языка нереально.
- Обычно декодер генерирует и поддерживает только лучшие гипотезы; это называется *beam search*.
- Т.е. мы после каждого шага (обычно на границах слов) делаем pruning и либо выкидываем гипотезы, которые сильно хуже текущего лучшего варианта, либо просто поддерживаем  $N$  лучших (типа 1000).

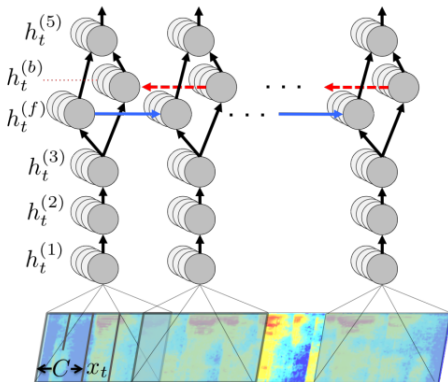
# Результаты

- А НММ для отдельных слов (нам же нужно по НММ на каждое слово) тоже можно организовать в дерево (префиксное).



# End-to-end

- Впрочем, сейчас уже всё по-другому.
- End-to-end speech recognition.
- Но об этом – потом.



Спасибо!

Спасибо за внимание!