ПРИМЕРЫ ПРИМЕНЕНИЯ АЛГОРИТМА EM

Сергей Николенко

СПбГУ — Санкт-Петербург

14 марта 2024 г.

Random facts:

- В Японии и Корее, как водится, всё наоборот: 14 февраля подарки дарят женщины (гири-тёко, «шоколад из чувства долга»), поэтому 14 марта — Белый день, когда мужчины делают ответные подарки (сначала маршмеллоу, потом белый шоколад)
- 14 марта — День числа пи; его обычно празднуют 3/14 в 1:59:26 дня; кстати, 14 марта родился Альберт Эйнштейн и умер Стивен Хокинг
- 14 марта — День православной книги, приуроченный к дню выпуска первой на Руси печатной книги Ивана Фёдорова «Апостол» (1 марта 1564 года по старому стилю)
- 14 марта 1930 г. одиннадцатилетняя школьница Венеция Бёрни предложила имя «Плутон» для открытой девятой планеты; 24 марта в обсерватории Лоуэлла это имя получило все голоса, и 1 мая было официально опубликовано
- 14 марта 1939 г. германские войска начали оккупацию Чехословакии; Словакия и Подкарпатская Русь формально провозгласили независимость, а единственным организованным сопротивлением немецкому вторжению был бой роты солдат под командованием капитана Павлика (бой за Чаянковы казармы)
- 14 марта 1994 г. состоялся релиз Linux версии 1.0.0

Пример EM: presence-only data

Presence-only data

- Пример из экологии: пусть мы хотим оценить, где водятся те или иные животные.
- Как определить, что суслики тут водятся, понятно: видишь суслика — значит, он есть.
- Но как определить, что суслика нет? Может быть, ты не видишь суслика, и я не вижу, а он есть?..

Presence-only data

- Формально говоря, есть переменные $\mathbf{x}$, определяющие некий регион (квадрат на карте), и мы моделируем вероятность того, что нужный вид тут есть, $p(y = 1 \mid \mathbf{x})$, при помощи логит-функции:

$$p(y = 1 \mid \mathbf{x}) = \sigma(\eta(\mathbf{x})) = \frac{1}{1 + e^{-\eta(\mathbf{x})}},$$

  где $\eta(\mathbf{x})$ может быть линейной (тогда получится логистическая регрессия), но может, в принципе, и не быть.
- Заметим, что даже если бы мы знали настоящие y, это было бы ещё не всё: сэмплирование положительных и отрицательных примеров неравномерно, перекошено в пользу положительных.
- *Важное замечание: prospective vs. retrospective studies, case-control studies.*

Presence-only data

- Это значит, что ещё есть пропорции сэмплирования (sampling rates)

$$\gamma_0 = p(s = 1 \mid y = 0), \quad \gamma_1 = p(s = 1 \mid y = 1),$$

  т.е. вероятности взять в выборку положительный/отрицательный пример.

- Их можно оценить как

$$\gamma_0 = \frac{n_0}{(1-\pi)N}, \quad \gamma_1 = \frac{n_1}{\pi N},$$

  где π — истинная доля положительных примеров (встречаемость, occurrence).

- Кстати, эту π было бы очень неплохо оценить в итоге.

Presence-only data

- Тогда, если знать истинные значения всех y, то в принципе можно обучить:

$$p(y = 1 \mid s = 1, \mathbf{x}) =$$

$$= \frac{p(s = 1 \mid y = 1, \mathbf{x})p(y = 1 \mid \mathbf{x})}{p(s = 1 \mid y = 0, \mathbf{x})p(y = 0 \mid \mathbf{x}) + p(s = 1 \mid y = 1, \mathbf{x})p(y = 1 \mid \mathbf{x})} =$$

$$= \frac{\gamma_1 e^{\eta(\mathbf{x})}}{\gamma_0 + \gamma_1 e^{\eta(\mathbf{x})}} = \frac{e^{\eta^*(\mathbf{x})}}{1 + e^{\eta^*(\mathbf{x})}},$$

где $\eta^*(\mathbf{x}) = \eta(\mathbf{x}) + \log(\gamma_1/\gamma_0)$, т.е.

$$\eta^*(\mathbf{x}) = \eta(\mathbf{x}) + \log\left(\frac{n_1}{n_0}\right) - \log\left(\frac{\pi}{1-\pi}\right).$$

Presence-only data

- Таким образом, если π неизвестно, то $\eta(\mathbf{x})$ можно найти с точностью до константы.
- А у нас не n_0 и n_1, а naive presence n_p и background n_u, т.е.

$$p(y=1 \mid s=1) = \frac{n_p+\pi n_u}{n_p+n_u}, \quad p(y=1 \mid s=0) = \frac{(1-\pi)n_u}{n_p+n_u},$$
$$\gamma_1 = \frac{p(y=1|s=1)p(s=1)}{p(y=1)} = \frac{n_p+\pi n_u}{\pi(n_p+n_u)}p(s=1),$$
$$\gamma_0 = \frac{p(y=0|s=1)p(s=1)}{p(y=0)} = \frac{n_u}{n_p+n_u}p(s=1),$$

  и в нашей модели $\log \frac{n_1}{n_0} = \log \frac{n_p+\pi n_u}{\pi n_u}$, т.е. всё как раньше, но $n_1 = n_p + \pi n_u$, $n_0 = (1-\pi)n_u$.
- Обучать по y можно так: обучить модель, а потом вычесть из $\eta(\mathbf{x})$ константу $\log \frac{n_p+\pi n_u}{\pi n_u}$.

Presence-only data

- Но у нас нет настоящих данных y, чтобы обучить регрессию, а есть только presence-only z: если $z = 1$, то $y = 1$, но если $z = 0$, то неизвестно, чему равен y.
- (Ward et al., 2009): давайте использовать EM. Правдоподобие:

$$\mathcal{L}(\eta \mid \mathbf{y}, \mathbf{z}, X) = \prod_i p(y_i, z_i \mid s_i = 1, \mathbf{x}_i) =$$

$$= \prod_i p(y_i \mid s_i = 1, \mathbf{x}_i) p(z_i \mid y_i, s_i = 1, \mathbf{x}_i).$$

- В нашем случае при n_p положительных примеров и n_u фоновых (неизвестных)

$$\begin{aligned} p(z_i = 0 \mid y_i = 0, s_i = 1, \mathbf{x}_i) &= 1, \\ p(z_i = 1 \mid y_i = 1, s_i = 1, \mathbf{x}_i) &= \frac{n_p}{n_p + \pi n_u}, \\ p(z_i = 0 \mid y_i = 1, s_i = 1, \mathbf{x}_i) &= \frac{\pi n_u}{n_p + \pi n_u}. \end{aligned}$$

Presence-only data

- А максимизировать нам надо сложное правдоподобие, в котором значения $\mathbf{y}$ неизвестны:

$$L(\eta \mid \mathbf{z}, X) = \prod_i p(z_i \mid s_i = 1, \mathbf{x}) =$$

$$= \prod_i \left(\frac{\frac{n_p}{\pi n_u} e^{\eta(\mathbf{x}_i)}}{1 + \left(1 + \frac{n_p}{\pi n_u}\right) e^{\eta(\mathbf{x}_i)}} \right)^{z_i} \left(\frac{1 + e^{\eta(\mathbf{x}_i)}}{1 + \left(1 + \frac{n_p}{\pi n_u}\right) e^{\eta(\mathbf{x}_i)}} \right)^{1 - z_i}.$$

- Для этого и нужен EM.

Presence-only data

- E-шаг здесь в том, чтобы заменить y_i на его оценку

$$\hat{y}_i^{(k)} = \mathbb{E}\left[y_i \mid \eta^{(k)}\right] = \frac{e^{\eta^{(k)}} + 1}{1 + e^{\eta^{(k)}} + 1}.$$

- M-шаг мы уже видели, это обучение параметров логистической модели с целевой переменной $\mathbf{y}^{(k)}$ на данных X.

(1) Chose initial estimates: $\hat{y}_i^{(0)} = \pi$ for $z_i = 0$.
(2) Repeat until convergence:

- *Maximization step:*
  - Calculate $\hat{\eta}^{*(k)}$ by fitting a logistic model of $\hat{\mathbf{y}}^{(k-1)}$ given X.
  - Calculate $\hat{\eta}^{(k)} = \hat{\eta}^{*(k)} - \log\left(\frac{n_p + \pi n_u}{\pi n_u}\right)$.
- *Expectation step:*

$$\hat{y}_i^{(k)} = \frac{e^{\hat{\eta}^{(k)}}}{1 + e^{\hat{\eta}^{(k)}}} \text{ for } z_i = 0 \quad \text{and} \quad \hat{y}_i^{(k)} = 1 \text{ for } z_i = 1$$

Presence-only data

- У Ward et al. получалось хорошо, но тут вышел любопытный спор.
- Ward et al. писали так: хотелось бы, чтобы можно было оценить π, но "π is identifiable only if we make unrealistic assumptions about the structure of $\eta(\mathbf{x})$ such as in logistic regression where $\eta(\mathbf{x})$ is linear in $\mathbf{x}$: $\eta(\mathbf{x}) = \mathbf{x}^\top \beta$".
- Через пару лет вышла статья Royle et al. (2012), тоже очень цитируемая, в которой говорилось: "logistic regression... is hardly unrealistic... such models are the most common approach to modeling binary variables in ecology (and probably all of statistics)... the logistic functions... is customarily adopted and widely used, and even books have been written about it".

Presence-only data

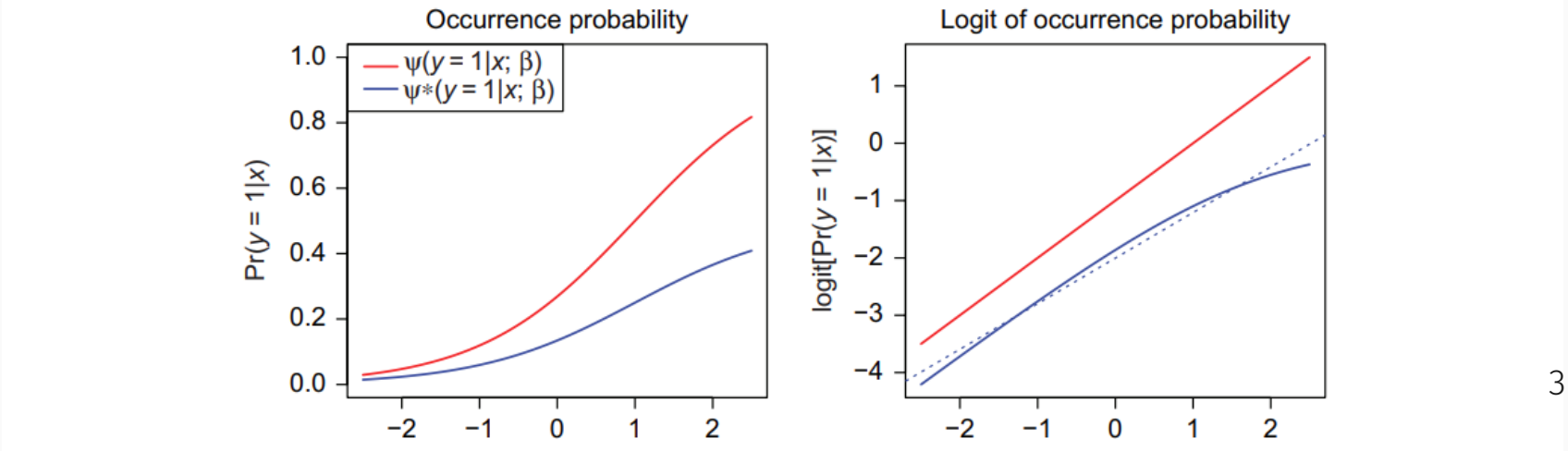
- Они предложили процедуру для оценки встречаемости π:
  - для признаков $\mathbf{x}$ у нас $p(y = 1 \mid \mathbf{x}) = \frac{p(y=1)\pi_1(\mathbf{x})}{p(y=1)\pi_1(\mathbf{x})+(1-p(y=1))\pi_0(\mathbf{x})}$;
  - данные – это выборка из $\pi_1(\mathbf{x})$ и отдельно выборка из $\pi(\mathbf{x})$;
  - как видно, даже если знать π и π_1 полностью, остаётся свобода: надо оценить $p(y = 1 \mid \mathbf{x})$, а $\pi_0(\mathbf{x})$ мы не знаем;
  - Royle et al. вводят предположения для $p(y = 1 \mid \mathbf{x})$ в виде логистической регрессии: $p(y = 1 \mid \mathbf{x}) = \sigma(\beta^\top \mathbf{x})$;
  - тогда действительно можно записать $p(y = 1)\pi_1(\mathbf{x}) = p(y = 1 \mid \mathbf{x})\pi(\mathbf{x}) = p(y = 1, \mathbf{x})$, и если $\pi(\mathbf{x})$ равномерно (это логично), то

$$\pi_1(\mathbf{x}_i) = \frac{p(y_i = 1 \mid \mathbf{x}_i)}{\sum_{\mathbf{x}} p(y = 1 \mid \mathbf{x})};$$

    если подставить сюда логистическую регрессию, то можно оценить β максимального правдоподобия, и не надо знать $p(y = 1)$!
- Что тут не так?

Presence-only data

- Всё так, но предположение о логистической регрессии здесь выполняет слишком много работы.
- Если рассмотреть две кривые, у которых $p^*(y = 1 \mid \mathbf{x}, \beta) = \frac{1}{2} p(y = 1 \mid \mathbf{x}, \beta)$, то у них будет $p^*(y = 1) = \frac{1}{2} p(y = 1)$, но общее правдоподобие $\pi_1(\mathbf{x}_i)$ будет в точности одинаковое, $\frac{1}{2}$ сократится.
- Дело в том, что модель p^* не будет логистической регрессией, с β произойдёт что-то нелинейное; но откуда у нас настолько сильное предположение? Как отличить синюю кривую справа от пунктирной прямой?



Пример: рейтинг спортивного ЧГК

Рейтинг спортивного ЧГК

- Пример из практики: в какой-то момент я хотел сделать рейтинг спортивного «Что? Где? Когда?»:
  - участвуют команды по ≤ 6 человек, причём часто встречаются неполные команды;
  - игроки постоянно переходят между командами (поэтому TrueSkill);
  - в одном турнире могут участвовать до тысячи команд (синхронные турниры);
  - командам задаётся фиксированное число вопросов (36, 60, 90), т.е. в крупных турнирах очень много команд делят одно и то же место.

Рейтинг спортивного ЧГК

- Первое решение — система TrueSkill
- Расскажу о ней потом, когда будем говорить об Expectation Propagation
- У неё есть проблемы в постановке спортивного ЧГК, мы когда-то их отчасти решили, была сложная и интересная модель, она работала лучше базового TrueSkill
- Но...

Рейтинг спортивного ЧГК

- В какой-то момент база турниров ЧГК стала собирать повопросные результаты.
- Мы теперь знаем, на какие именно вопросы ответила та или иная команда.
- Так что когда я вернулся к задаче построения рейтинга ЧГК, задача стала существенно проще.

Рейтинг спортивного ЧГК

- Пример аналогичного приложения:
  - есть набор вопросов для теста из большого числа вопросов (например, IQ-тест или экзамен по какому-то предмету);
  - участники отвечают на случайное подмножество вопросов;
  - надо оценить участников, но уровень сложности вопросов нельзя заранее точно сбалансировать.
- «Что? Где? Когда?» — это оно и есть, только теперь участники объединяются в команды.

Baseline: логистическая регрессия

- Baseline — логистическая регрессия:
  - каждый игрок i моделируется скиллом s_i,
  - каждый вопрос q моделируется сложностью («лёгкостью») c_q,
  - добавим глобальное среднее μ,
  - обучим логистическую модель

$$p(x_{tq} \mid s_i, c_q) \sim \sigma(\mu + s_i + c_q)$$

для каждого игрока $i \in t$ команды-участницы $t \in T^{(d)}$ и каждого вопроса $q \in Q^{(d)}$, где $\sigma(x) = 1/(1 + e^x)$ — логистический сигмоид, x_{tq} — ответила ли команда t на вопрос q.

Модель со скрытыми переменными

- Логистическая модель предполагает фактически, что каждый игрок ответил на каждый вопрос, который взяла команда.
- Это неправда, мы не знаем, кто ответил, только знаем, что кто-то это сделал; а если команда не ответила, то никто.
- Это по идее похоже на presence-only data models (Ward et al., 2009; Royle et al., 2012).

Модель со скрытыми переменными

- Поэтому давайте сделаем модель со скрытыми переменными.
- Для каждой пары игрок-вопрос, добавим переменную z_{iq}, которая означает, что «игрок i ответил на вопрос q».
- На эти переменные есть такие ограничения:
  - если $x_{tq} = 0$, то $z_{iq} = 0$ для каждого игрока $i \in t$;
  - если $x_{tq} = 1$, то $z_{iq} = 1$ для по крайней мере одного игрока $i \in t$.

Модель со скрытыми переменными

- Параметры модели те же — скилл и сложность вопросов:

$$p(z_{iq} \mid s_i, c_q) \sim \sigma(\mu + s_i + c_q).$$

- Обучаем EM-алгоритмом:
  - E-шаг: зафиксируем все s_i и c_q, вычислим ожидания скрытых переменных z_{iq} как

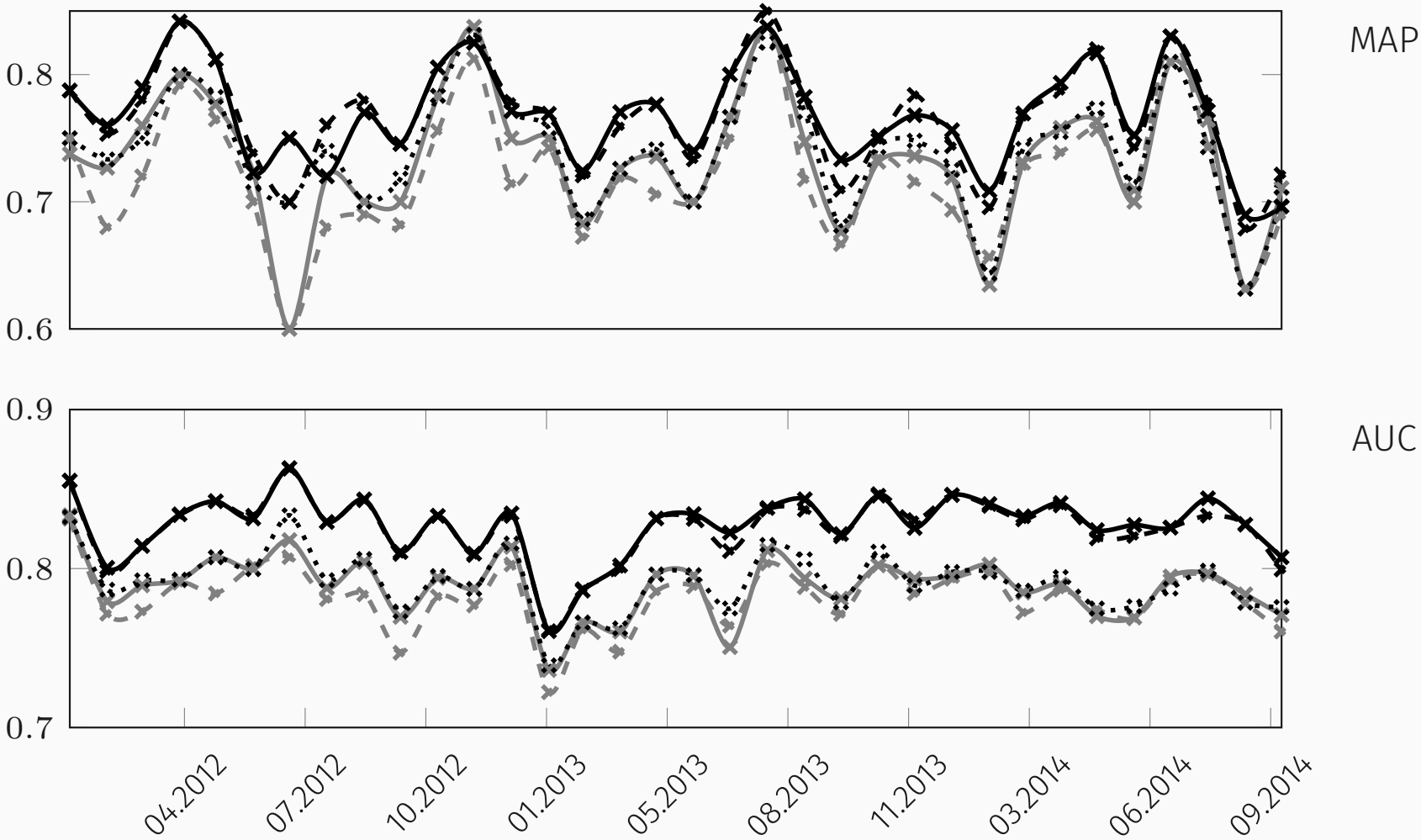
$$\mathbb{E}\left[z_{iq}\right] = \begin{cases} 0, & \text{если } x_{tq} = 0, \\ p(z_{iq} = 1 \mid \exists j \in t \; z_{jq} = 1) = \frac{\sigma(\mu + s_i + c_q)}{1 - \prod_{j \in t}\left(1 - \sigma(\mu + s_j + c_q)\right)}, & \text{если } x_{tq} = 1; \end{cases}$$

  - M-шаг: зафиксируем $\mathbb{E}\left[z_{iq}\right]$, обучим логистическую модель

$$\mathbb{E}\left[z_{iq}\right] \sim \sigma(\mu + s_i + c_q).$$

Результаты

- Ну и, конечно, работает хорошо.



Пример

Рейтинг ЧГК | ИГРОКИ | КОМАНДЫ | ТУРНИРЫ | ВОПРОСЫ | Рейтинг-лист игроков | РЕЛИЗ: ЯНВАРЬ 2015

РЕЙТИНГ-ЛИСТ ИГРОКОВ НА ЯНВАРЬ 2015

Показывать по 100 записей

Введите id или начало фамилии:

Место	id	Фамилия	Имя	Отчество	Команда	Сыграно	Взято	Рейтинг
1	27177	Ромашова	Вероника	Михайловна	ЛКИ	5278	3784	545.753
2	3083	Белявский	Дмитрий	Михайлович	ЛКИ	4831	3396	543.520
3	27403	Руссо	Максим	Михайлович	ЛКИ	7180	5205	534.540
4	4270	Брутер	Александра	Владимировна	ЛКИ	8244	5974	533.405
5	18332	Либер	Александр	Витальевич	Рабочее название	8658	6210	532.921
6	1585	Архангельская	Юлия	Сергеевна	Ксеп	7542	5286	531.866
7	24384	Пашковский	Евгений	Александрович	ЛКИ	7552	5374	531.327
8	8333	Губанов	Антон	Александрович	Команда Губанова	4475	3153	530.871
9	16332	Крапиль	Николай	Валерьевич	Ксеп	6927	4899	530.559
10	21487	Моносов	Борис	Яковлевич	Команда Губанова	5115	3645	530.175

Пример

Рейтинг ЧГК ИГРОКИ КОМАНДЫ ТУРНИРЫ ВОПРОСЫ Александр Друзь vs. Максим Поташев

Ссылка на сайт рейтинга МАК

Друзь
Александр Абрамович

Поташев
Максим Оскарович

Ссылка на сайт рейтинга МАК

ИСТОРИЯ РЕЙТИНГА | ИСТОРИЯ ТУРНИРОВ

ИСТОРИЯ РЕЙТИНГОВ

Рейтинг	Команда	Место	Дата	Место	Команда	Рейтинг
450.138	Транссфера	303	Январь 2015	31	Афина	514.643
446.669	Транссфера	336	Декабрь 2014	29	Афина	514.697
442.559	Транссфера	348	Ноябрь 2014	25	Афина	513.155
437.999	Транссфера	400	Октябрь 2014	25	Афина	512.083
445.791	Транссфера	306	Сентябрь 2014	23	Афина	515.737
441.473	Транссфера	372	Август 2014	23	Афина	516.513
446.730	Транссфера	284	Июль 2014	22	Афина	516.755
448.598	Транссфера	322	Июнь 2014	24	Афина	516.603

Скрытые марковские модели: основное

Марковские цепи

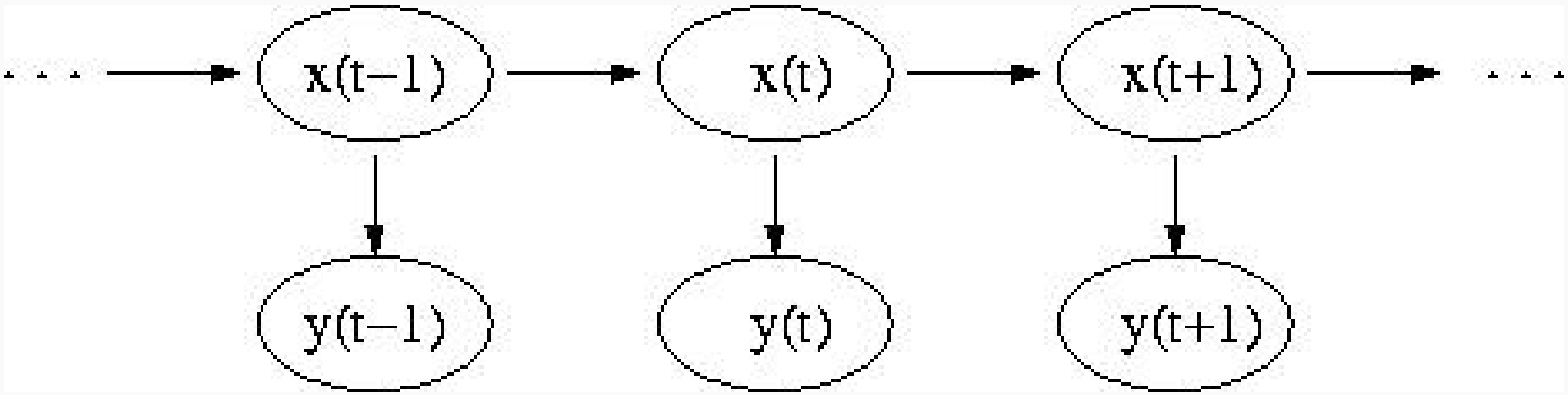
- Марковская цепь задаётся начальным распределением вероятностей $p^0(x)$ и вероятностями перехода $T(x'; x)$.
- $T(x'; x)$ — это распределение следующего элемента цепи в зависимости от следующего; распределение на $(t+1)$–м шаге равно

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

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

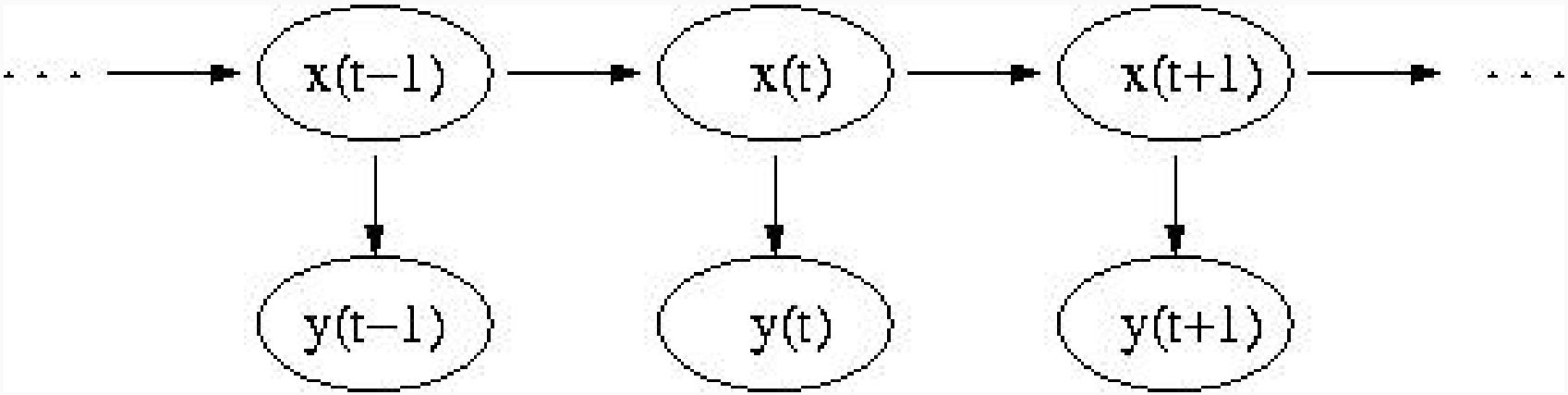
Дискретные марковские цепи

- 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.

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

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

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).
- 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).

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 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

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

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

- Ничего не напоминает?

Суть решения первой задачи

- Правильно, это такая же задача маргинализации, как мы решаем всё время.
- Мы воспользуемся 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?

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

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

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:

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

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

- 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 (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:

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

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

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

- 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.

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 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 Kullback–Leibler:

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

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

Вспомогательная функция

- Введём вспомогательную функцию

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

- Тогда из неравенства следует, что

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

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

Функция Q

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

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

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

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

- 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!).

Специальные виды марковских моделей

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 μ и вариацией σ (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}, μ_{jm} и σ_{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:

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

Вычисление $\alpha_t(i)$

- Тогда получается

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

Вычисление $\alpha_t(i)$

- 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,

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

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.
- π_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 α есть ещё и β, 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:

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

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

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

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

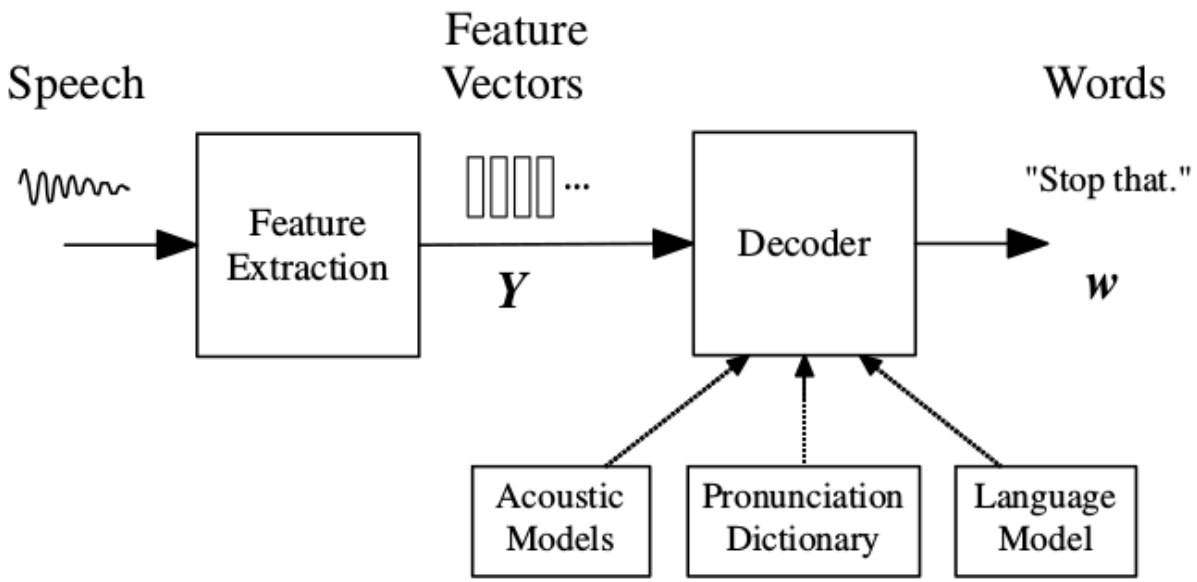
- Чтобы уменьшить количество параметров, можно иногда считать, что $p_i(d)$ — классическое распределение с не слишком большим количеством параметров.
- Например, $p_i(d)$ может быть равномерным, или нормальным ($p_i(d) = \mathcal{N}(d, \mu_i, \sigma_i^2)$), или гамма–распределением:

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

HMM для распознавания речи

Общая структура

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



От слов к фонемам

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

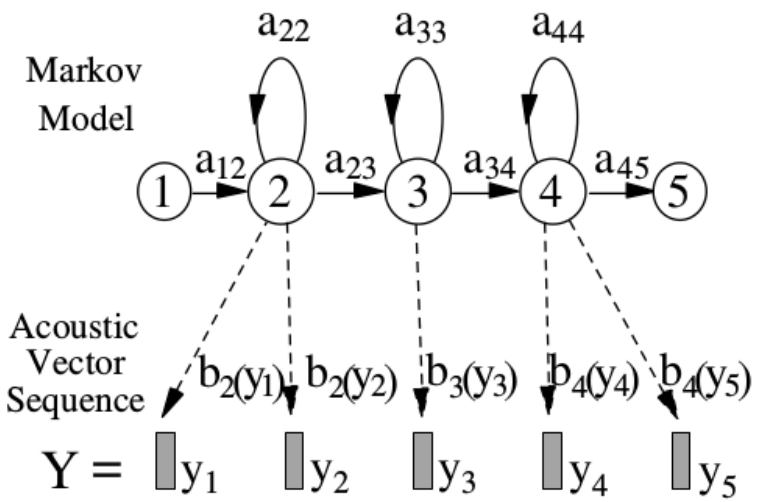
$$p(\mathbf{y} \mid \mathbf{w}) = \sum_{\mathbf{q}} p(\mathbf{y} \mid \mathbf{q}) p(\mathbf{q} \mid \mathbf{w}),$$

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

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

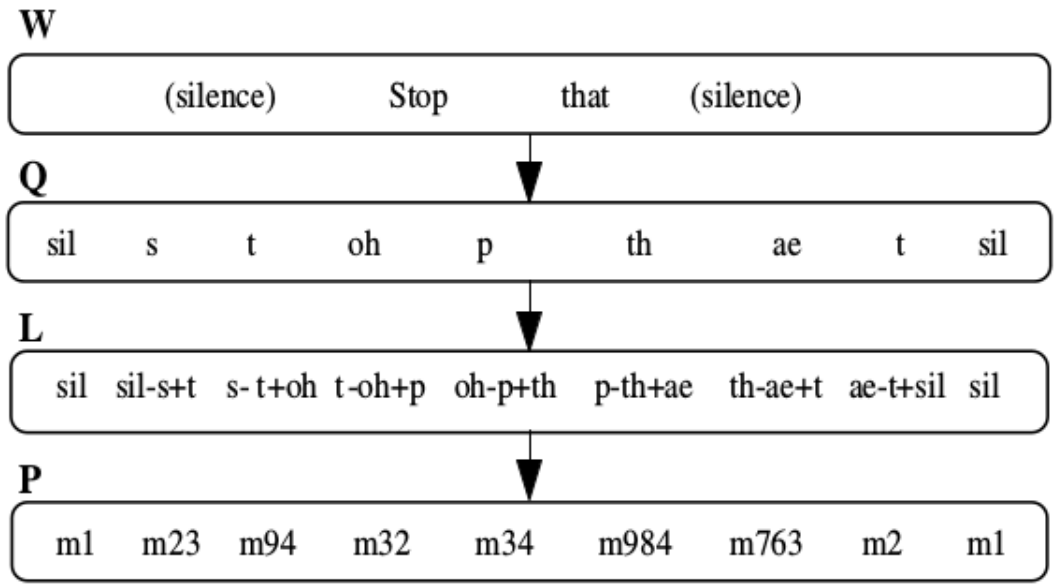
Акустическая модель

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



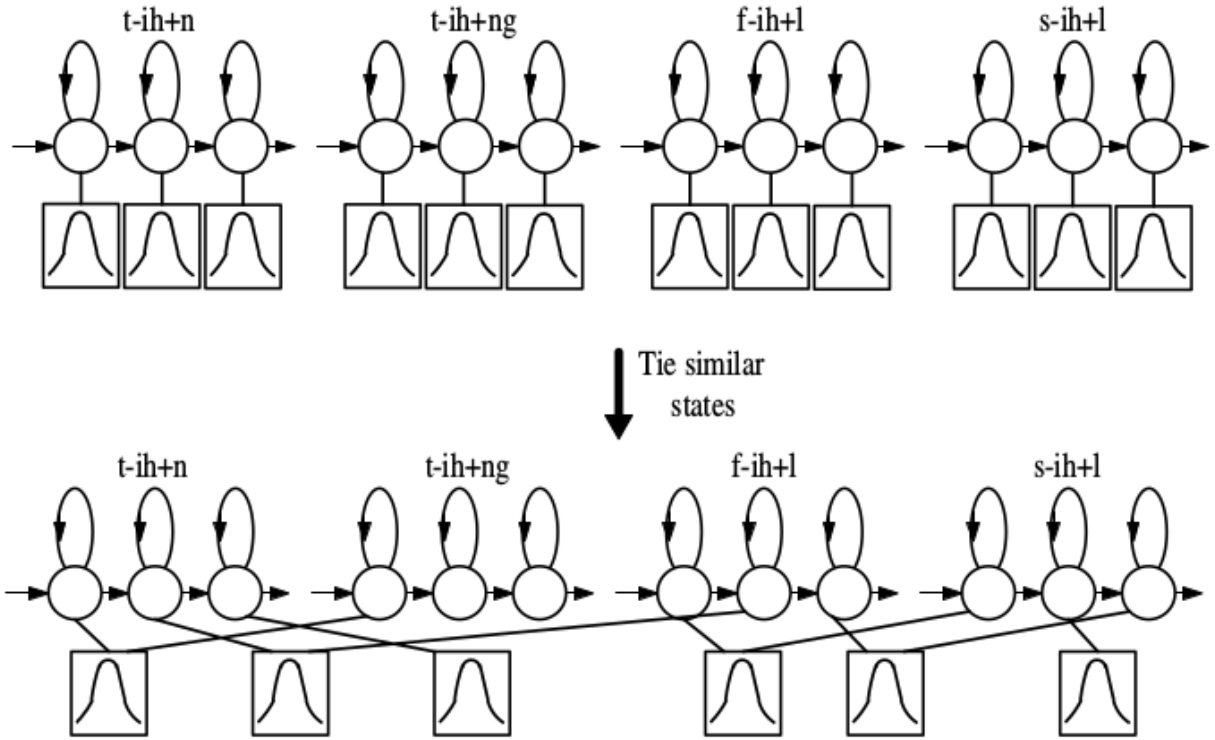
Акустическая модель

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



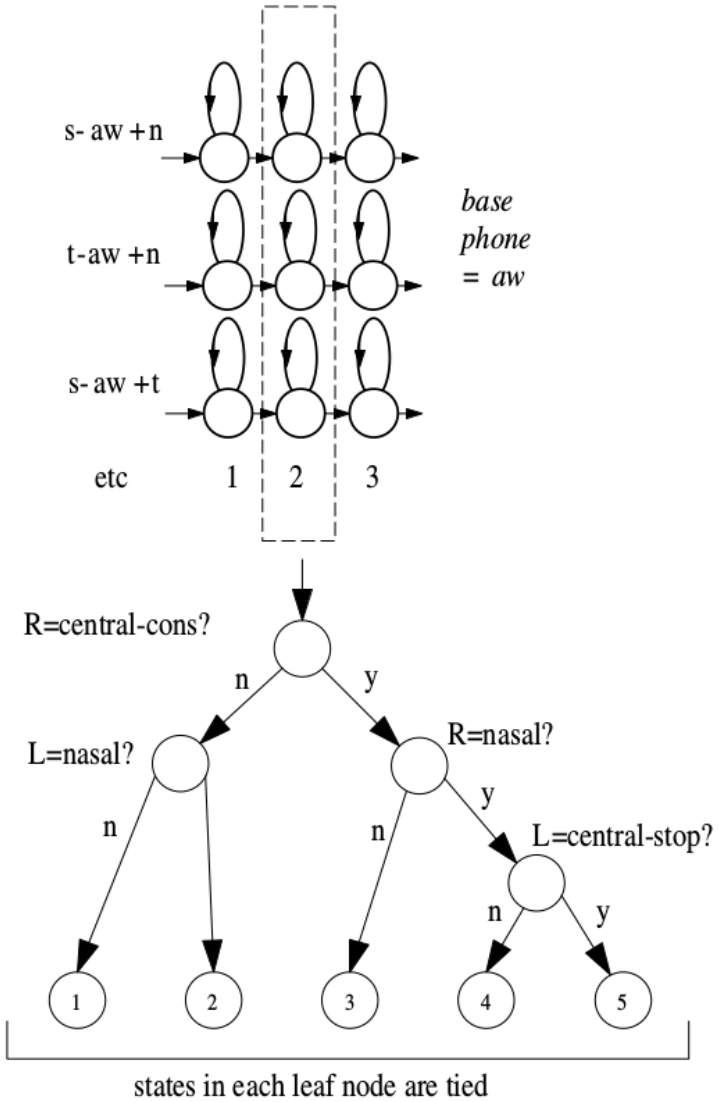
Акустическая модель

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



Акустическая модель

- Это можно сделать просто силой мысли:



Языковая модель

- 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)

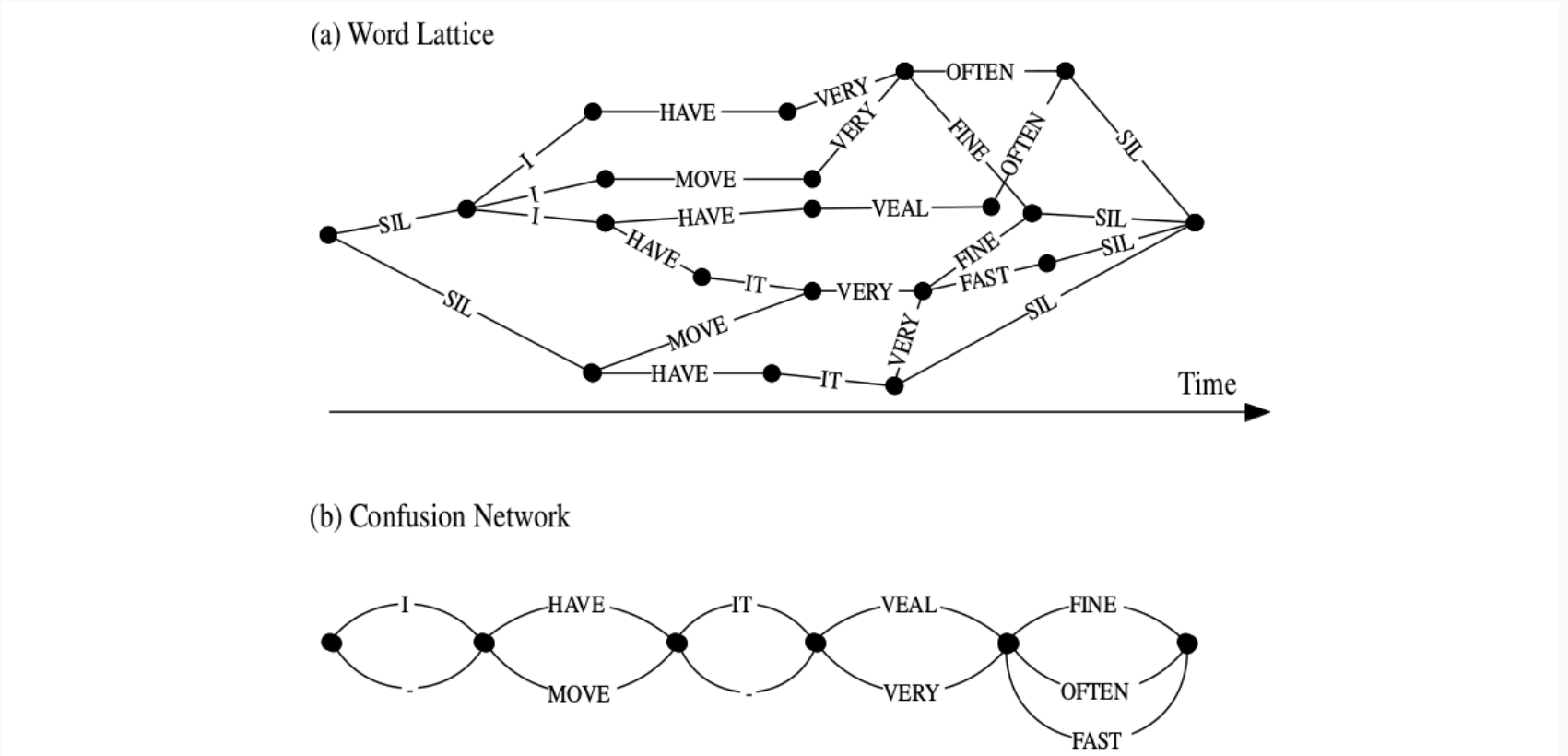
$$H = -\lim_{L\to\infty} \frac{1}{K} \log p(w_1, \dots, w_L).$$

Результаты

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

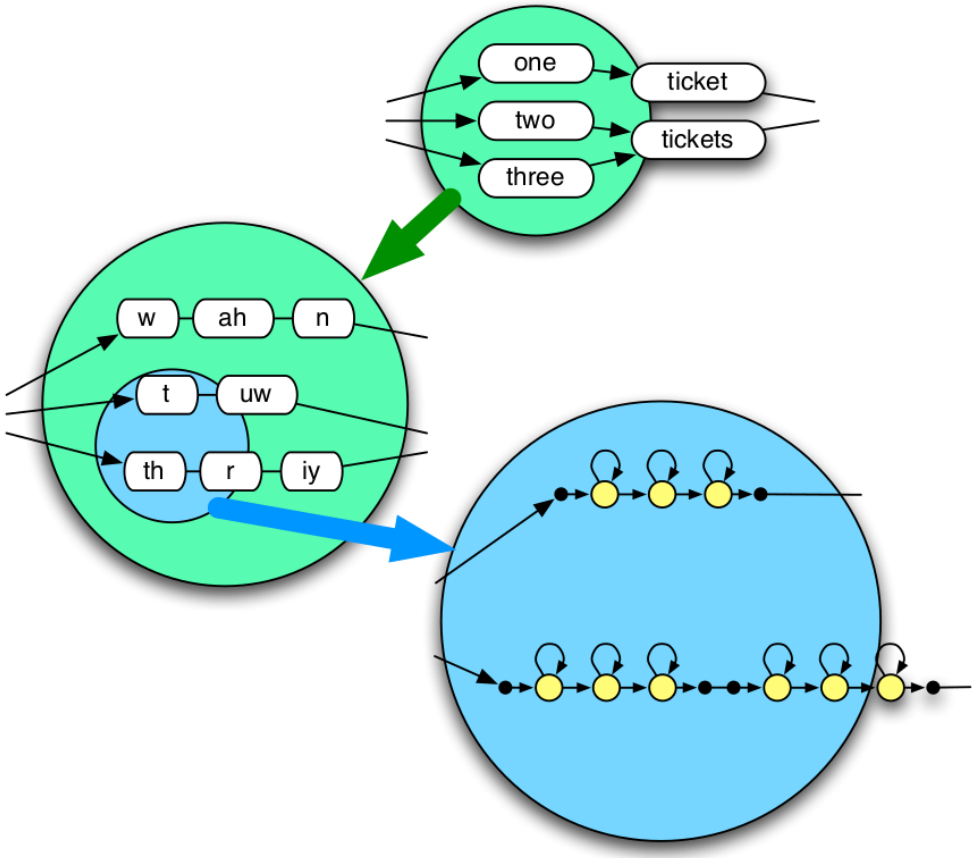
Результаты

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



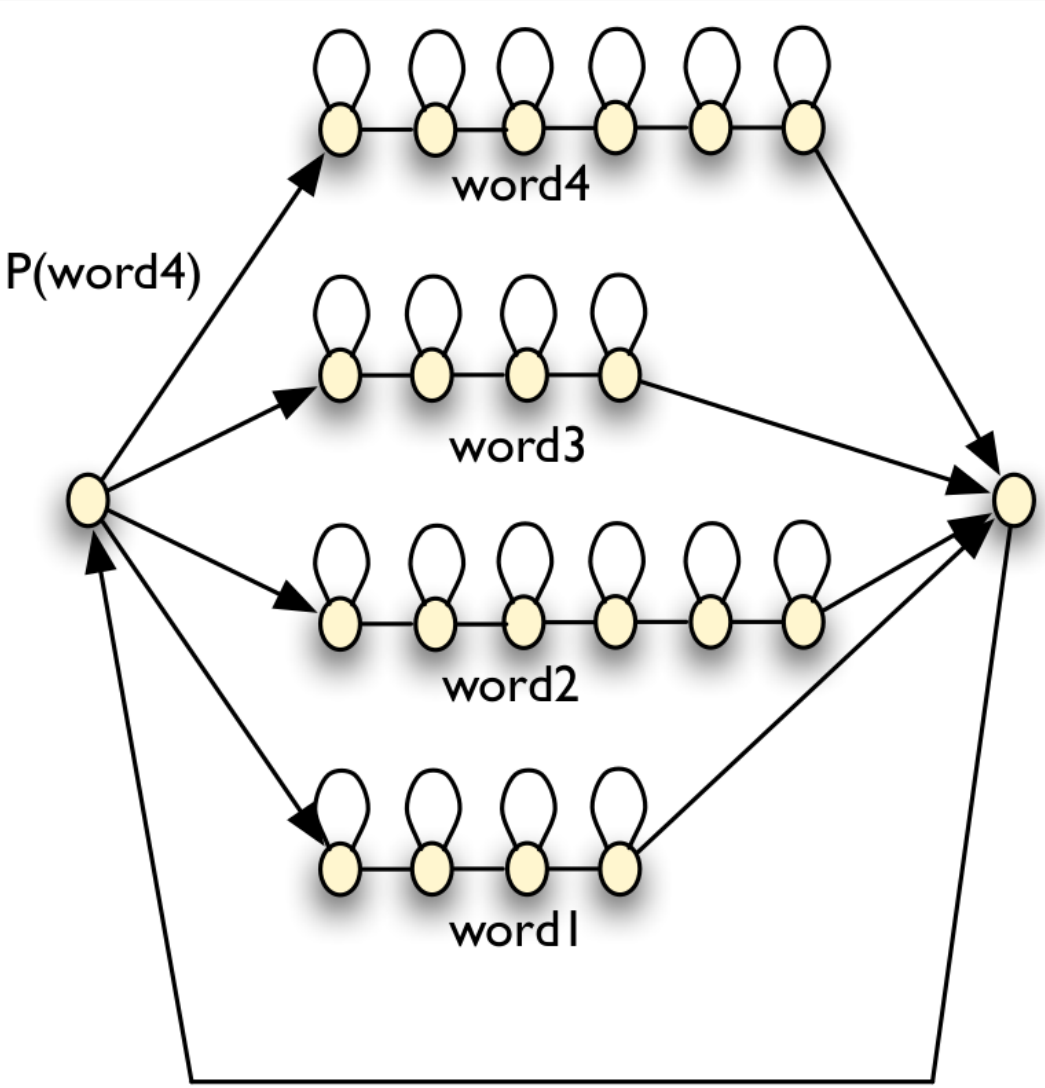
Результаты

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



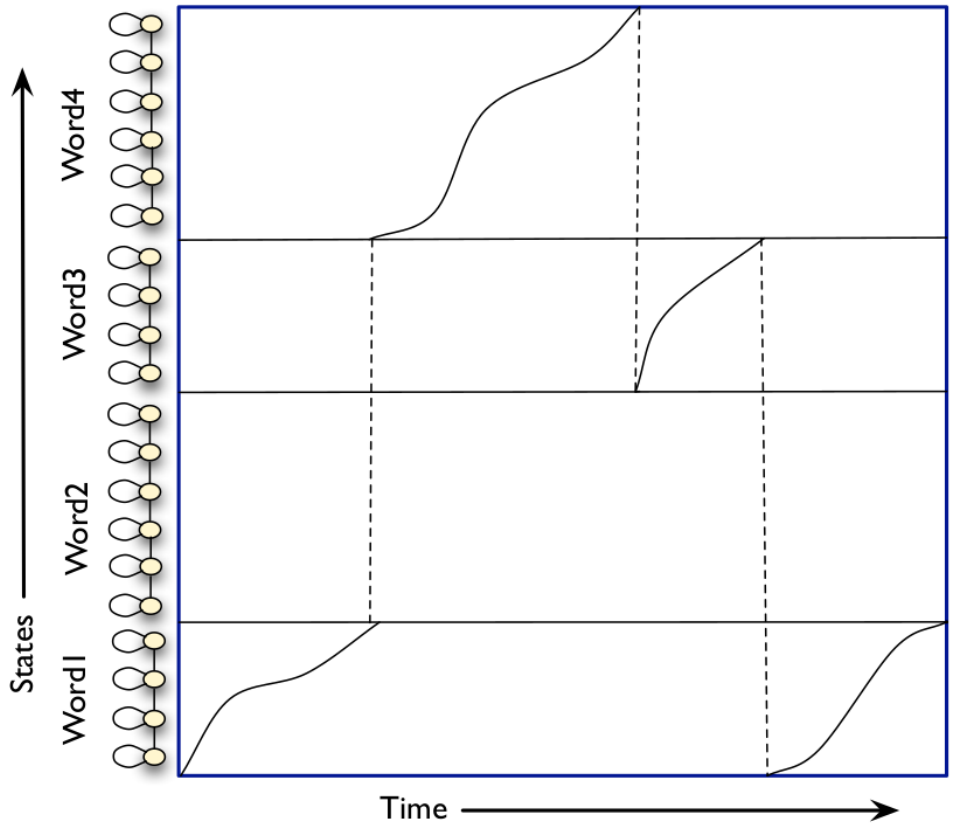
Результаты

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



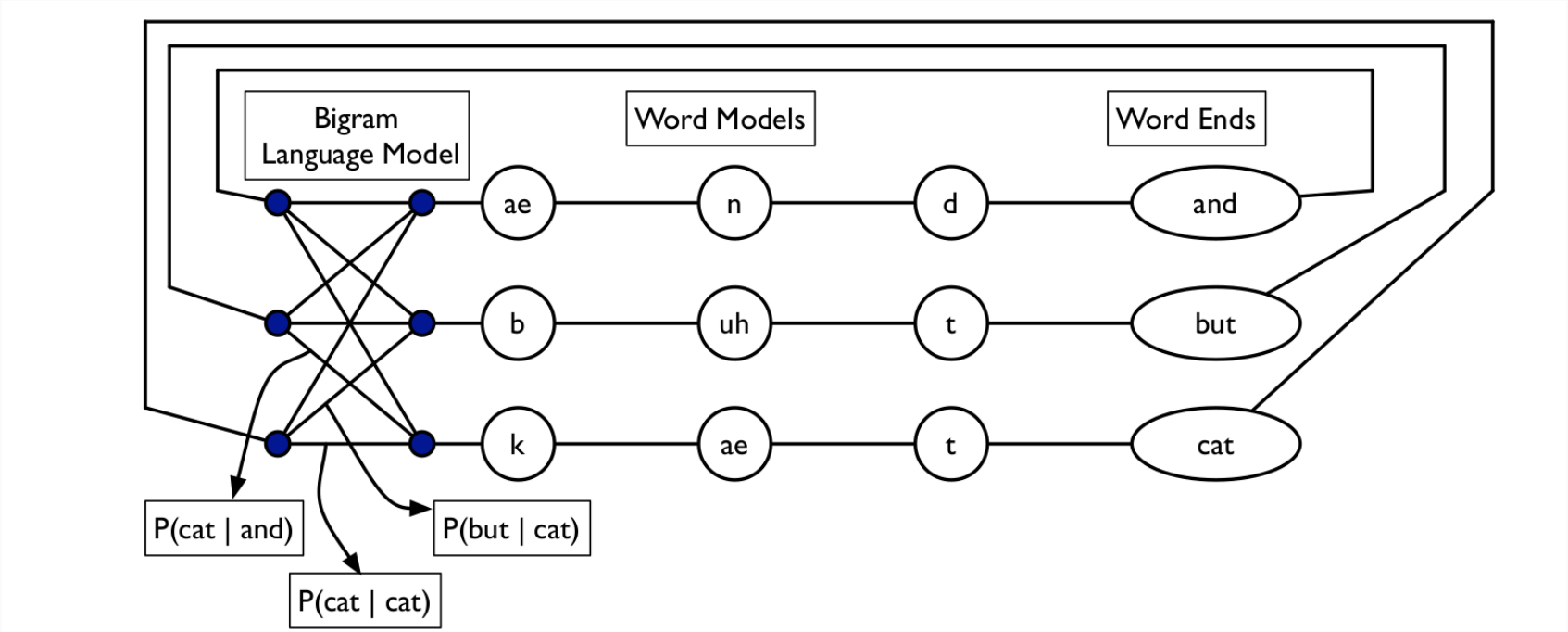
Результаты

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



Результаты

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

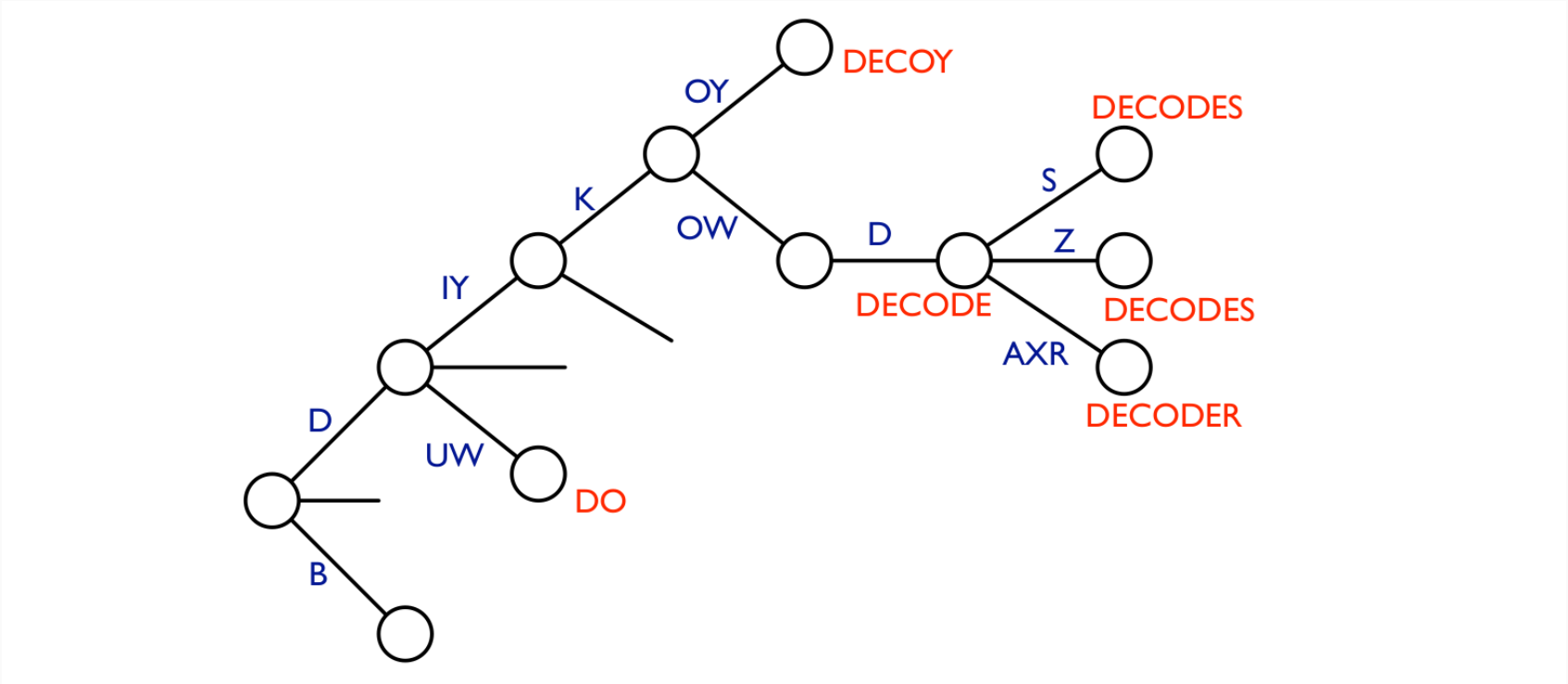


Результаты

- 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).

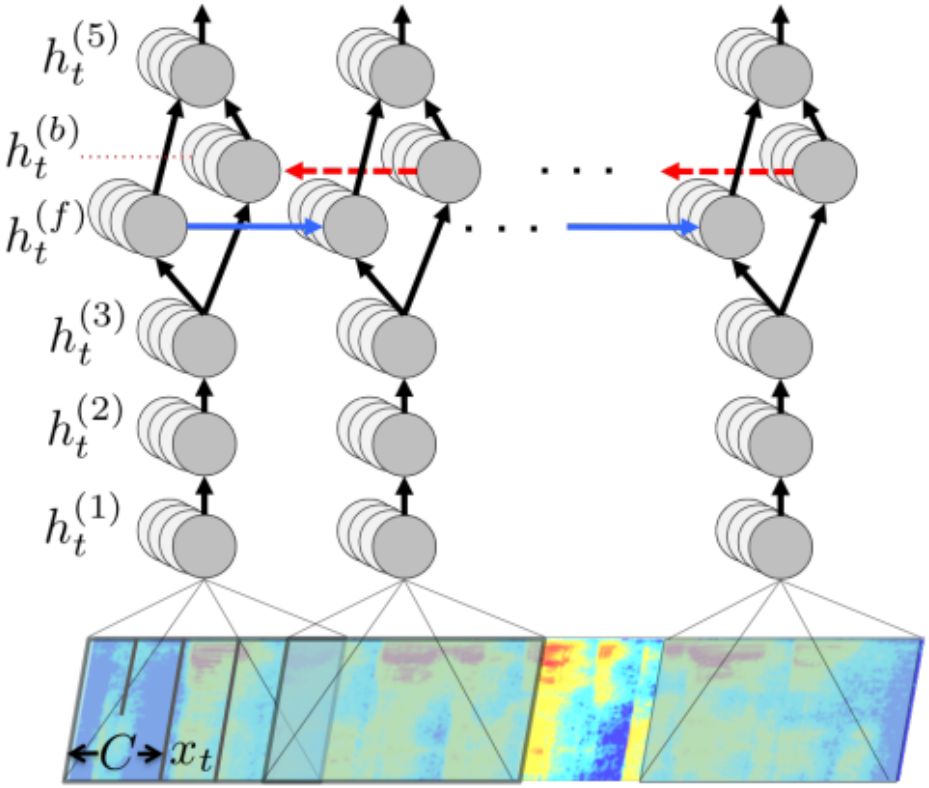
Результаты

- А HMM для отдельных слов (нам же нужно по HMM 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!