

Обработка речевых сигналов

Блок 2. Автоматическое распознавание речи

Максим Корневский
Старший научный сотрудник ООО «ЦРТ»,
к.ф.-м.н.



Настоящий блок лекций подготовлен при
поддержке «ЦРТ | Группа компаний»



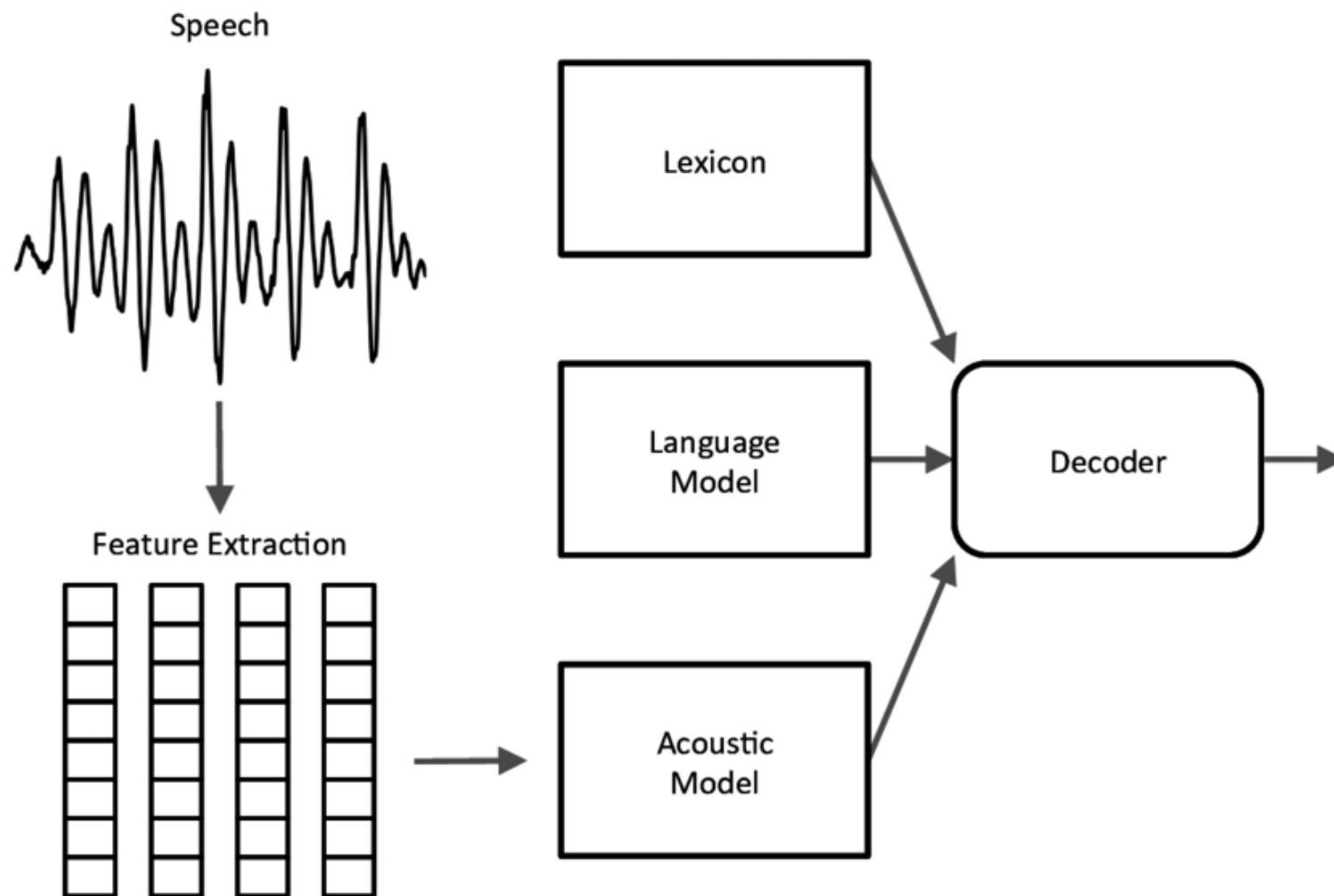
Блок 2. Автоматическое распознавание речи (Automatic Speech Recognition, ASR)



Часть 3. Системы распознавания речи на основе GMM-HMM



Напоминание: Архитектура (традиционной) ASR-системы



План лекции

- Скрытые Марковские Модели и связанные с ними задачи
- Применение НММ для распознавания речи
- Гауссовы смеси, обучение GMM-НММ
- Графы распознавания. Взвешенные конечные преобразователи. WFST-декодер. Словные сети
- Дискриминативное обучение GMM-НММ
- Адаптация систем распознавания речи

План лекции

- Скрытые Марковские Модели и связанные с ними задачи
- Применение НММ для распознавания речи
- Гауссовы смеси, обучение GMM-НММ
- Графы распознавания. Взвешенные конечные преобразователи. WFST-декодер. Словные сети
- Дискриминативное обучение GMM-НММ
- Адаптация систем распознавания речи

Скрытые Марковские модели и связанные задачи

Скрытая Марковская модель (Hidden Markov Model, НММ)

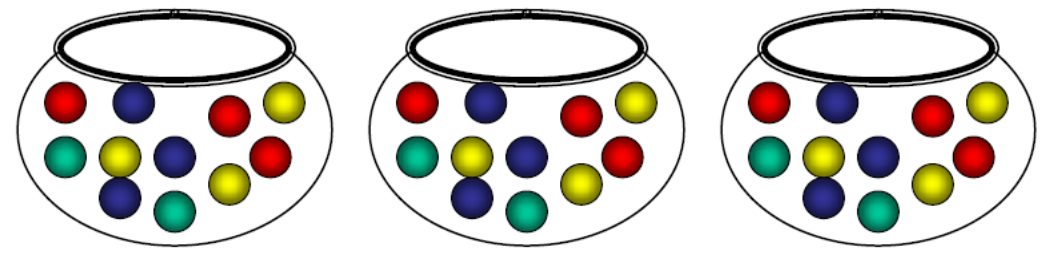
- **Скрытая** Марковская модель: $\lambda = (\pi, A, B)$
 - $\pi_i = P(q_1 = i)$, $i = 0, 1, \dots, N$ - начальные вероятности,
 - $a_{ij} = P(q_t = S_j | q_{t-1} = S_i)$ – вероятности переходов
 - Состояния процесса **не наблюдаются** непосредственно
 - Есть множество наблюдаемых значений V
 - В каждом состоянии задано вероятностное **распределение** наблюдаемых в нем значений:
 $V = \{v_1, v_2, \dots, v_M\}$ - множество значений, $b_{jk} = P(o_t = v_k | q_t = S_j)$ - наборы вероятностей,
 - B – набор этих вероятностных распределений: $B = \{b_0, b_1, \dots, b_N\}$
 - Распределения могут быть и **непрерывными**: $b_j(o_t) = p(o_t | q_t = S_j)$ - плотности, а не вероятности
- Отдельные наблюдения при фиксированных состояниях **НЕЗАВИСИМЫ** (frame independence assumption)

Скрытые Марковские модели и связанные задачи

HMM – генеративная модель

- HMM может использоваться для генерации последовательности наблюдений $O = (o_1, o_2, \dots, o_T)$:
 1. Положить $t = 1$
 2. Выбрать начальное состояние $q_1 = i$ в соответствии с распределением $\{\pi_i\}$
 3. Сгенерировать наблюдение $o_t = v_k$ в соответствии с распределением $\{b_i\}$ в текущем состоянии
 4. Выбрать следующее состояние $q_{t+1} = j$ в соответствии с распределением $\{a_{ij}\}$
 5. Положить $t = t + 1$ и перейти к шагу 3, если $t \leq T$
- Примерно так работает «аппарат» в примере с урнами

Скрытая Марковская модель



Пример с урнами и шарами (напоминание)

- Есть 3 урны, в каждой из которых определенное известное количество шаров красного, синего и зеленого цвета. И есть некий «аппарат»:
 1. Вначале аппарат выбирает урну наугад в соответствии с некими вероятностями π_i
 2. После этого аппарат достает из урны случайно выбранный шар, записывает его цвет и возвращает обратно
 3. После этого аппарат выбирает, к какой урне переместиться согласно распределению a_{ij}
 4. Шаги 2-3 повторяются некоторое количество раз
- Наблюдатель видит только последовательность цветов, записанную аппаратом. Номера урн он не знает! Хочется уметь отвечать на вопросы:
 - Какова вероятность выбранной последовательности цветов?
 - Какой последовательности урн она наиболее вероятно соответствует?
 - Если содержимое урн и вероятности π_i и a_{ij} **неизвестны**, как их оценить по имеющимся записанным последовательностям цветов? Т.е. как **обучить** модель, используя наблюдения ?

Скрытые Марковские модели и связанные задачи

1. Вероятность последовательности наблюдений

- Есть последовательность наблюдений $O = (o_1, o_2, \dots, o_T)$. Как найти $P(O|\lambda)$?
- Рассмотрим всевозможные последовательности состояний $q = (q_1, q_2, \dots, q_T)$

$$P(q|\lambda) = \pi_{q_1} \cdot a_{q_1 q_2} \cdot a_{q_2 q_3} \cdots a_{q_{T-1} q_T}$$

- Вероятность наблюдений на заданной последовательности состояний

$$P(O|q, \lambda) = \prod_{t=1}^T P(o_t|q_t, \lambda) = b_{q_1}(o_1) \cdot b_{q_2}(o_2) \cdots b_{q_T}(o_T)$$

- По формуле полной вероятности:

$$P(O|\lambda) = \sum_q P(O, q|\lambda) = \sum_q P(O|q, \lambda)P(q|\lambda) = \sum_q \pi_{q_1} b_{q_1}(o_1) a_{q_1 q_2} b_{q_2}(o_2) a_{q_2 q_3} \cdots a_{q_{T-1} q_T} b_{q_T}(o_T).$$

Скрытые Марковские модели и связанные задачи

Forward-алгоритм

- Введем вспомогательную величину (**forward-вероятность**): $\alpha_t(i) = P(o_1 o_2 \dots o_t, q_t = i | \lambda)$
- Несложно понять, что ее можно вычислять рекуррентно:

$$\alpha_1(i) = \pi_i b_i(o_1)$$
$$\alpha_{t+1}(j) = \left[\sum_{i=1}^N \alpha_t(i) a_{ij} \right] b_j(o_{t+1})$$

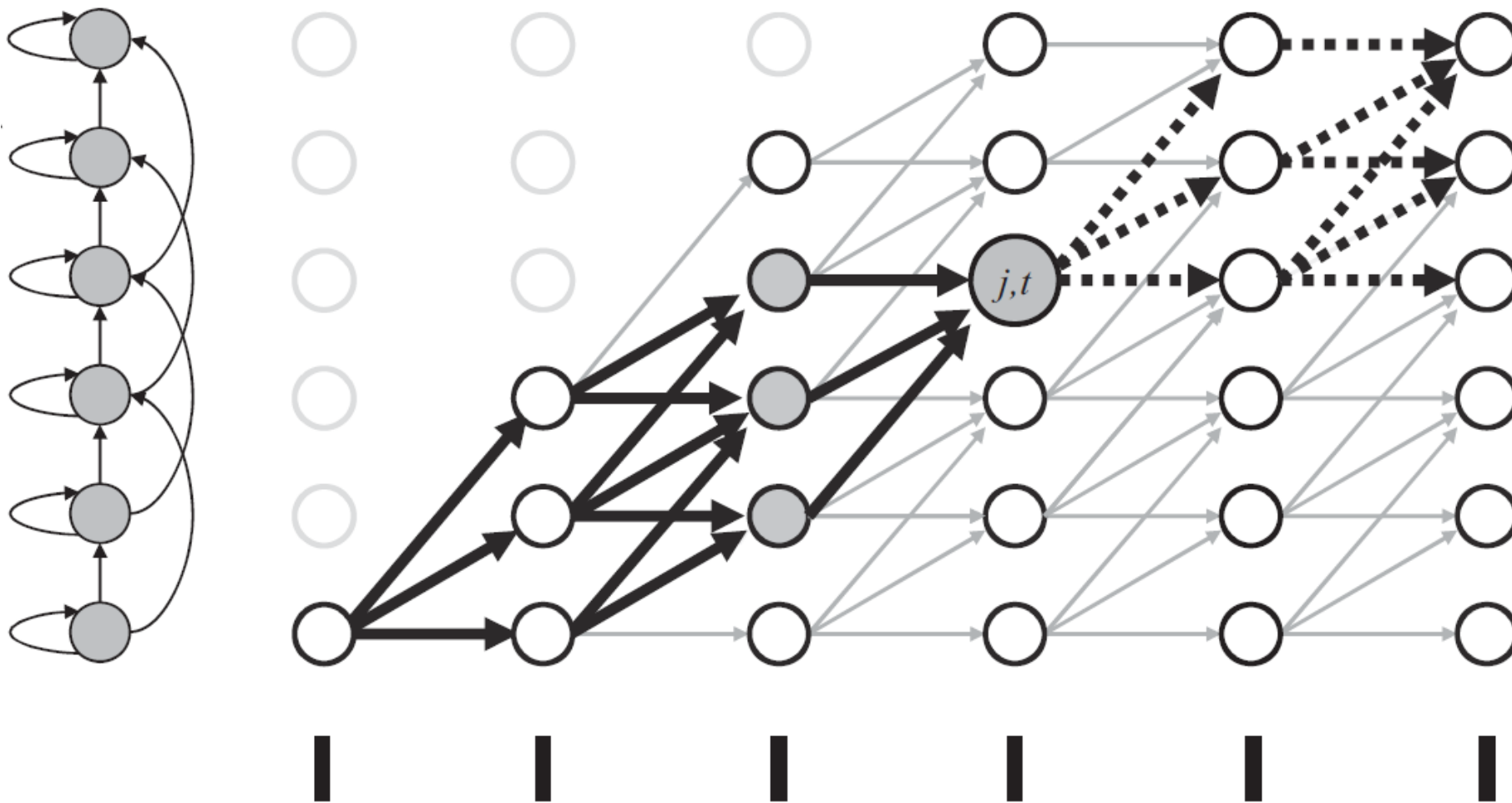
- Если вычислены значения $\alpha_T(i)$ во всех состояниях, то

$$P(O | \lambda) = \sum_{i=1}^N \alpha_T(i).$$

- Этот алгоритм позволяет вычислять $P(O | \lambda)$ за $O(TN^2)$ операций

Скрытые Марковские модели и связанные задачи

Forward-алгоритм: иллюстрация



Скрытые Марковские модели и связанные задачи

Backward-алгоритм (аналог в обратную сторону)

- Введем аналогичную величину (**backward-вероятность**): $\beta_t(i) = P(o_{t+1}o_{t+2} \dots o_T | q_t = i, \lambda)$
- Ее тоже можно вычислять рекуррентно:

$$\beta_T(i) = 1, \quad \beta_t(i) = \sum_{j=1}^N a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), \quad t = 0, \dots, T-1, i = 1, \dots, N$$

$$P(O|\lambda) = \sum_{i=1}^N \pi_i b_i(o_1) \beta_0(i).$$

- С помощью forward и backward-вероятностей можно вычислить вероятности прохода через данное состояние в данный момент времени:

$$\begin{aligned} P(O, q_t = i | \lambda) &= P(o_1 o_2 \dots o_T, q_t = i | \lambda) = \\ &= P(o_1 o_2 \dots o_t, q_t = i | \lambda) P(o_{t+1} o_{t+2} \dots o_T | q_t = i, \lambda) = \alpha_t(i) \beta_t(i) \end{aligned}$$

Скрытые Марковские модели и связанные задачи

2. Вычисление наилучшей последовательности состояний:

- Есть последовательность наблюдений $O = (o_1, o_2, \dots, o_T)$. На какой последовательности состояний $\hat{q} = (q_1, q_2, \dots, q_T)$ она наиболее вероятно наблюдается?

$$\hat{q} = \operatorname{argmax}_q P(O, q | \lambda)$$

- Динамическое программирование: определим вспомогательную величину

$$\delta_t(i) = \max_{q_1 q_2 \dots q_{t-1}} P(q_1 q_2 \dots q_{t-1}, q_t = i, o_1 o_2 \dots o_t | \lambda).$$

- Ее можно пересчитывать рекуррентно (**алгоритм Витерби**, A.Viterbi, 1972):

$$\delta_1(i) = \pi_i b_i(o_1), \quad \delta_{t+1}(j) = \max_i (\delta_t(i) a_{ij}) b_j(o_{t+1})$$

- Если на каждом шаге запоминать из какого состояния $\varphi_t(i)$ мы пришли в данное, то можно восстановить оптимальную последовательность состояний (**выравнивание**).
- Эту процедуру еще называют **forced alignment**.

Скрытые Марковские модели и связанные задачи

Алгоритм Витерби (более подробно):

- **Инициализация:** $\delta_1(i) = \pi_i b_i(o_1), \quad \varphi_1(i) = 0, \quad i = 1, 2, \dots, N$
- **Рекурсия:**
$$\delta_t(j) = \max_i (\delta_{t-1}(i) a_{ij}) b_j(o_t),$$
$$\varphi_t(j) = \operatorname{argmax}_i (\delta_{t-1}(i) a_{ij}) b_j(o_t), \quad j = 1, 2, \dots, N, \quad t = 2, \dots, T$$
- **Завершение:** $P^* = \max_i (\delta_T(i)), \quad \hat{q}_T = \operatorname{argmax}_i (\delta_T(i))$
- **Обратный ход** (восстановление последовательности состояний): $\hat{q}_t = \varphi_{t+1}(\hat{q}_{t+1})$
- NB: Лучше все вычисления производить в логарифмах (произведения \rightarrow суммы)

Скрытые Марковские модели и связанные задачи

3. Обучение Скрытой Марковской модели

- Пусть наша НММ должна описывать конкретное слово и есть набор «эталонов» этого слова. Как найти наилучшие параметры модели?
- **Метод максимального правдоподобия**: выбрать такие параметры, для которых достигается максимальная суммарная вероятность на эталонах

$$\hat{\lambda} = \operatorname{argmax}_{\lambda} P(O|\lambda)$$

- Нет эффективного алгоритма для поиска глобального максимума ☹️
- Приходится использовать итерационные подходы. Два самых распространенных:
 - Витерби-обучение
 - Алгоритм Баума-Уэлша (Baum-Welch algorithm).
 - Оба варианта - разновидности **ЕМ-алгоритма**. На каждой итерации происходит обновление параметров модели: $\lambda \rightarrow \bar{\lambda}$, при котором вероятность гарантированно **НЕ УБЫВАЕТ!**

Скрытые Марковские модели и связанные задачи

Витерби обучение:

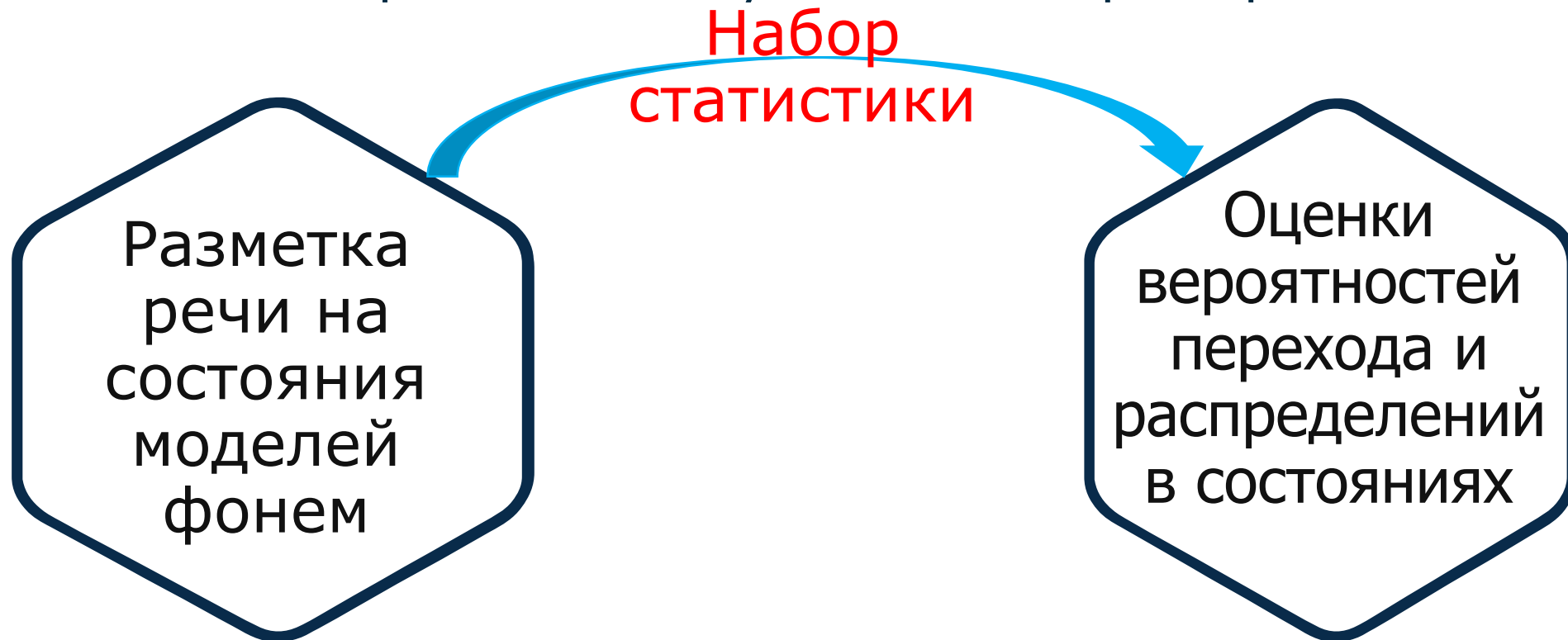
- Предположим, что у нас есть разметка наблюдений на состояния НММ:



Скрытые Марковские модели и связанные задачи

Витерби обучение:

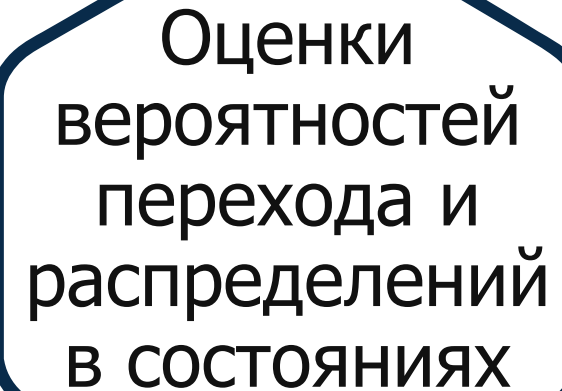
- Тогда мы можем набрать статистику и оценить параметры НММ:



Скрытые Марковские модели и связанные задачи

Витерби обучение:

- Предположим теперь, что у нас есть данные и описывающая их НММ:

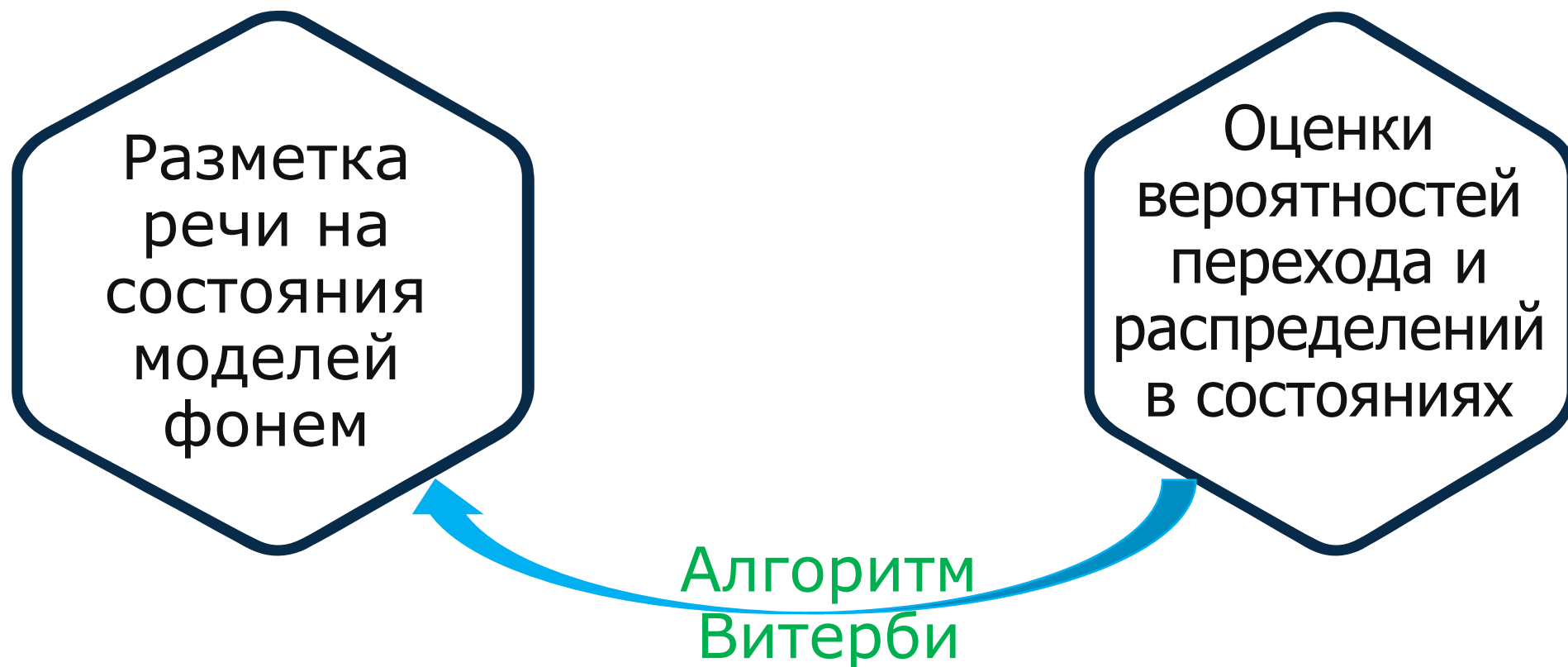


Оценки
вероятностей
перехода и
распределений
в состояниях

Скрытые Марковские модели и связанные задачи

Витерби обучение:

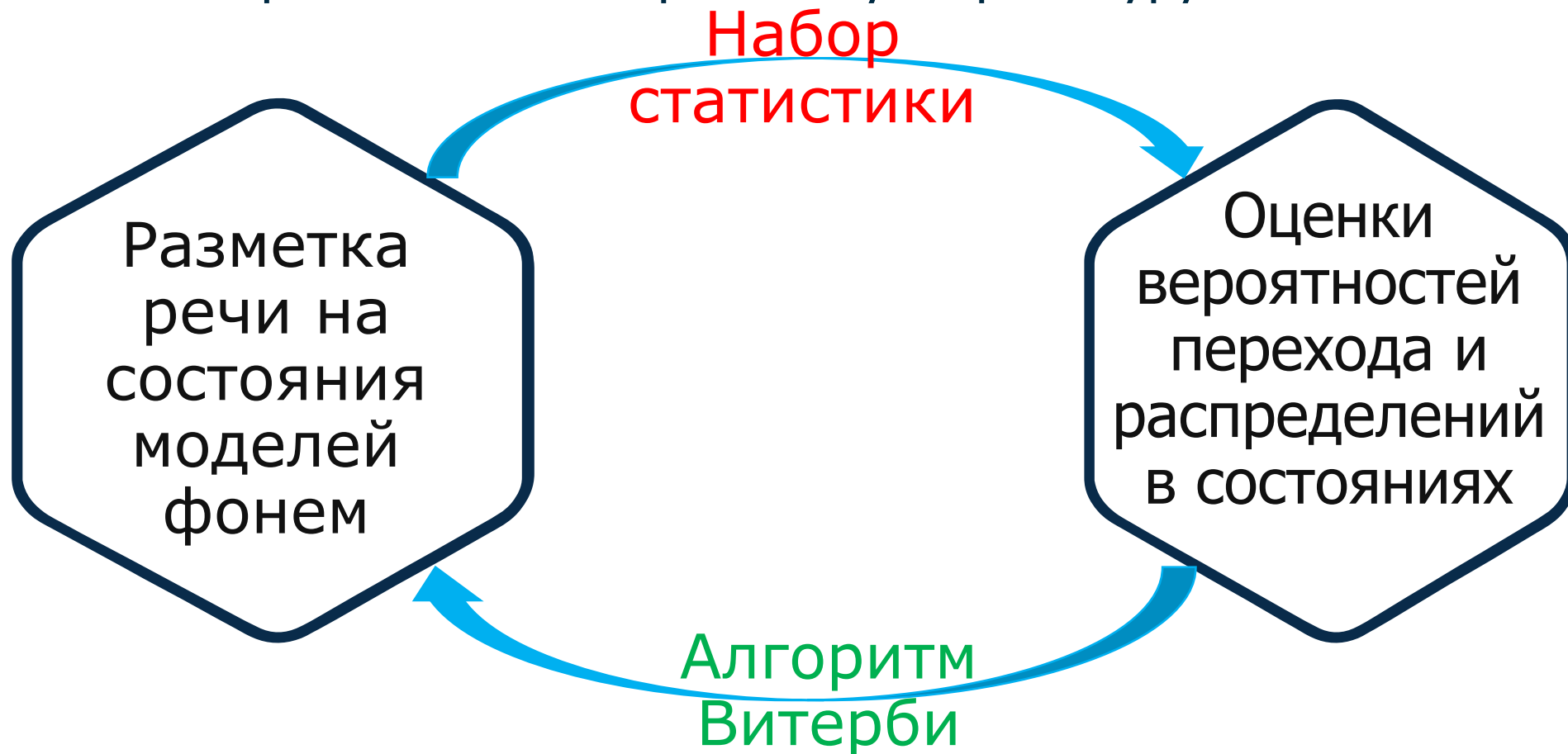
- Тогда мы можем найти наилучшее выравнивание данных алгоритмом Витерби (**forced alignment**, каждому наблюдению сопоставим состояние НММ):



Скрытые Марковские модели и связанные задачи

Витерби обучение:

- Значит можно организовать итерационную процедуру:



Скрытые Марковские модели и связанные задачи

Алгоритм Баума-Уэлша (Е-шаг):

- Используя текущие значения параметров модели λ , вычисляются вероятности:
 - Вероятность в момент времени t находиться в состоянии s_i на данной последовательности наблюдений (state occupancy probability):

$$\gamma_i(t) = P(s(t) = s_i \mid o_1, o_2, \dots, o_T; \lambda)$$

- Вероятность в момент времени t находиться в состоянии s_i , а в момент времени $(t + 1)$ - в состоянии s_j :

$$\xi_{ij}(t) = P(s(t) = s_i, s(t + 1) = s_j \mid o_1, o_2, \dots, o_T; \lambda)$$

- Их можно вычислить, используя forward и backward-вероятности:

$$\gamma_i(t) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_j \alpha_t(j)\beta_t(j)}$$

$$\xi_{ij}(t) = \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{P(O|\lambda)} = \frac{\gamma_i(t)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\beta_t(i)}$$

Скрытые Марковские модели и связанные задачи

Алгоритм Баума-Уэлша (М-шаг) для дискретной НММ:

- На М-шаге происходит обновление параметров НММ:

$$\pi_i = \gamma_1(i), \quad a_{ij} = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}$$

$$b_{jk} = b_j(v_k) = \frac{\sum_{t: o_t=v_k} \gamma_j(t)}{\sum_t \gamma_j(t)}$$

- Если эталонов, по которым строится НММ, несколько (N), то вероятности $\gamma_i(t)$ и $\xi_{ij}(t)$ вычисляются для каждого эталона в отдельности, а в формулы добавляется суммирование по всем эталонам:

$$\pi_i = \frac{\gamma_1^{(n)}(i)}{N}, \quad a_{ij} = \frac{\sum_n \sum_{t=1}^{T-1} \xi_{ij}^{(n)}(t)}{\sum_n \sum_{t=1}^{T-1} \gamma_i^{(n)}(t)}, \quad b_{jk} = \frac{\sum_n \sum_{t: o_t=v_k} \gamma_j^{(n)}(t)}{\sum_n \sum_t \gamma_j^{(n)}(t)}$$

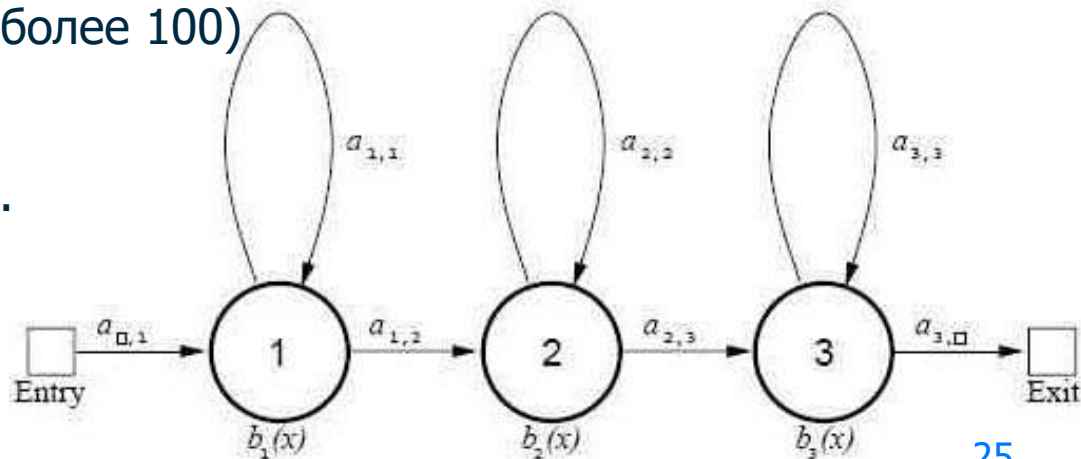
План лекции

- Скрытые Марковские Модели и связанные с ними задачи
- **Применение НММ для распознавания речи**
- Гауссовы смеси, обучение GMM-НММ
- Графы распознавания. Взвешенные конечные преобразователи. WFST-декодер. Словные сети
- Дискриминативное обучение GMM-НММ
- Адаптация систем распознавания речи

Применение НММ для распознавания речи

Недостатки построения НММ для отдельных слов

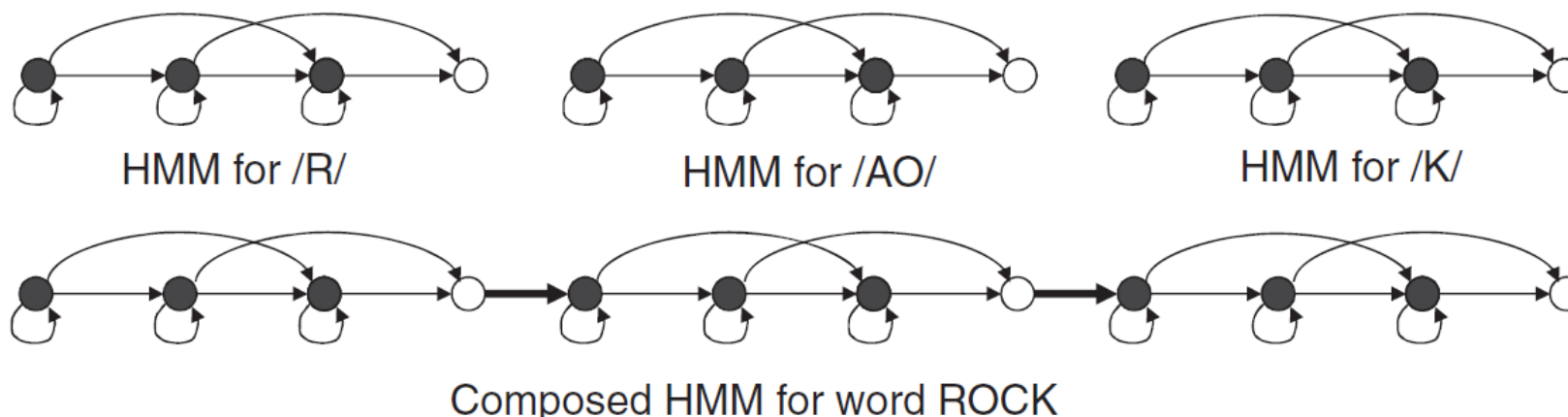
- Много состояний, много параметров («тяжелые» модели)
- С ростом размера словаря становится слишком затратным
- На каждое слово в обучающих данных может быть слишком мало примеров, как следствие – переобучение
- Выход: перейти на суб-словные единицы: слоги, **фонемы**.
 - Фонем в большинстве языков относительно немного (не более 100)
 - На каждую фонему значительно больше статистики
 - «Топология» НММ для фонем может быть очень простой.
 - Типичный вариант фонемной НММ: 3 state, left-to-right



Применение НММ для распознавания речи

Фонемные НММ и способы их улучшения

- Как учить фонемные НММ?
 - Фонемные НММ «склеиваются» в словные НММ, а они – в НММ для целой фразы



- Одинаковые состояния одинаковых фонем «разделяют» (share) общие параметры
 - При обучении надо накапливать статистики для состояний по всем вхождениям
- Типичное количество состояний: 150-200
- Основная проблема – в разном окружении фонемы сильно различаются: фонема «т» в слове «вата» совсем не такая, как в слове «строить». Это называется **коартикуляцией**

Применение НММ для распознавания речи

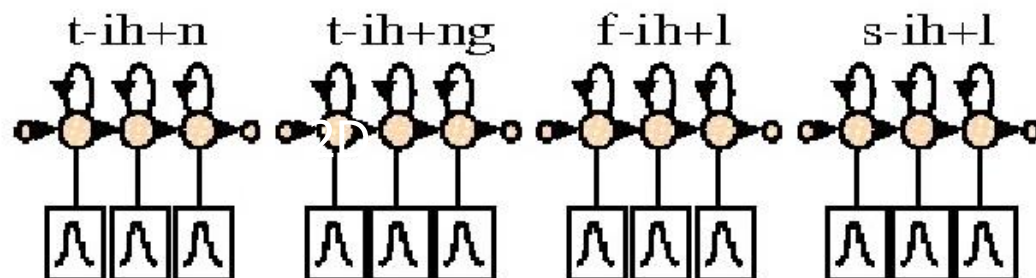
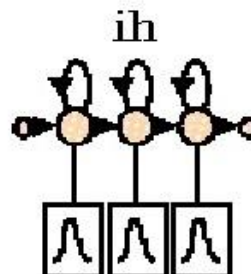
Учет фонетического контекста

- Фонема в определенном «окружении» называется **аллофоном**.
- Аллофон с контекстом в 1 фонему слева и справа называется **трифоном**.
- Фонемная запись: вата => v a0 t a4 (a0 – ударная, a4 – на конце слова)
- Трифонная запись: вата => v+a0 v-a0+t a0-t+a4 t-a4:
 - Трифон v-a0+t – это аллофон фонемы a0, которая находится в окружении v слева и t справа
 - Дифон v+a0 – это аллофон фонемы v справа от которой стоит фонема a0
 - Дифон t-a4 – это аллофон фонемы a4, слева от которой стоит фонема t
- При склеивании словных НММ следует учитывать влияние «межсловного» контекста («город Москва» vs. «город Санкт-Петербург»)
- Используют также «пентафоны» (или **quinphones**) - аллофоны с контекстом 2 фонемы с каждой стороны

Применение НММ для распознавания речи

Учет фонетического контекста

- Фонема в определенном «окружении» называется **аллофоном**.
- Аллофон с контекстом в 1 фонему слева и справа называется **трифоном**.

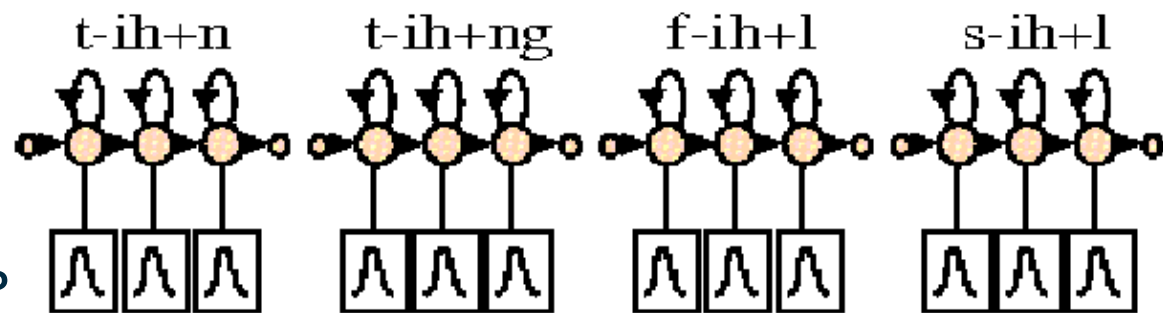


Применение НММ для распознавания речи

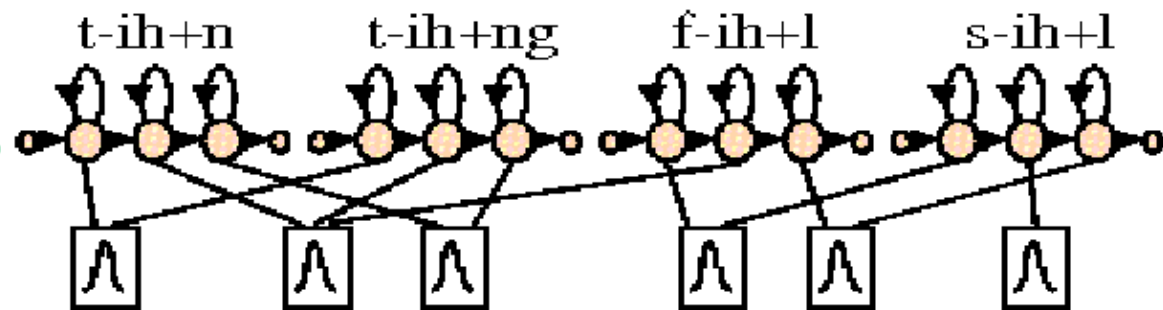
Проблемы трифонных моделей

- Различных трифонов очень много. Если фонем 50, то трифонов 125000.
- Очень многие трифоны из полного набора вообще никогда не встречаются в речи, очень многих в обучающих данных **нет вообще или очень мало!**
- Чтобы сохранить число параметров в разумных пределах придумали **связывать (tie)** похожие состояния трифонов. Связанные состояния (**сеноны, senones**) **разделяют (share)** общее распределение в состоянии.

Conventional triphones



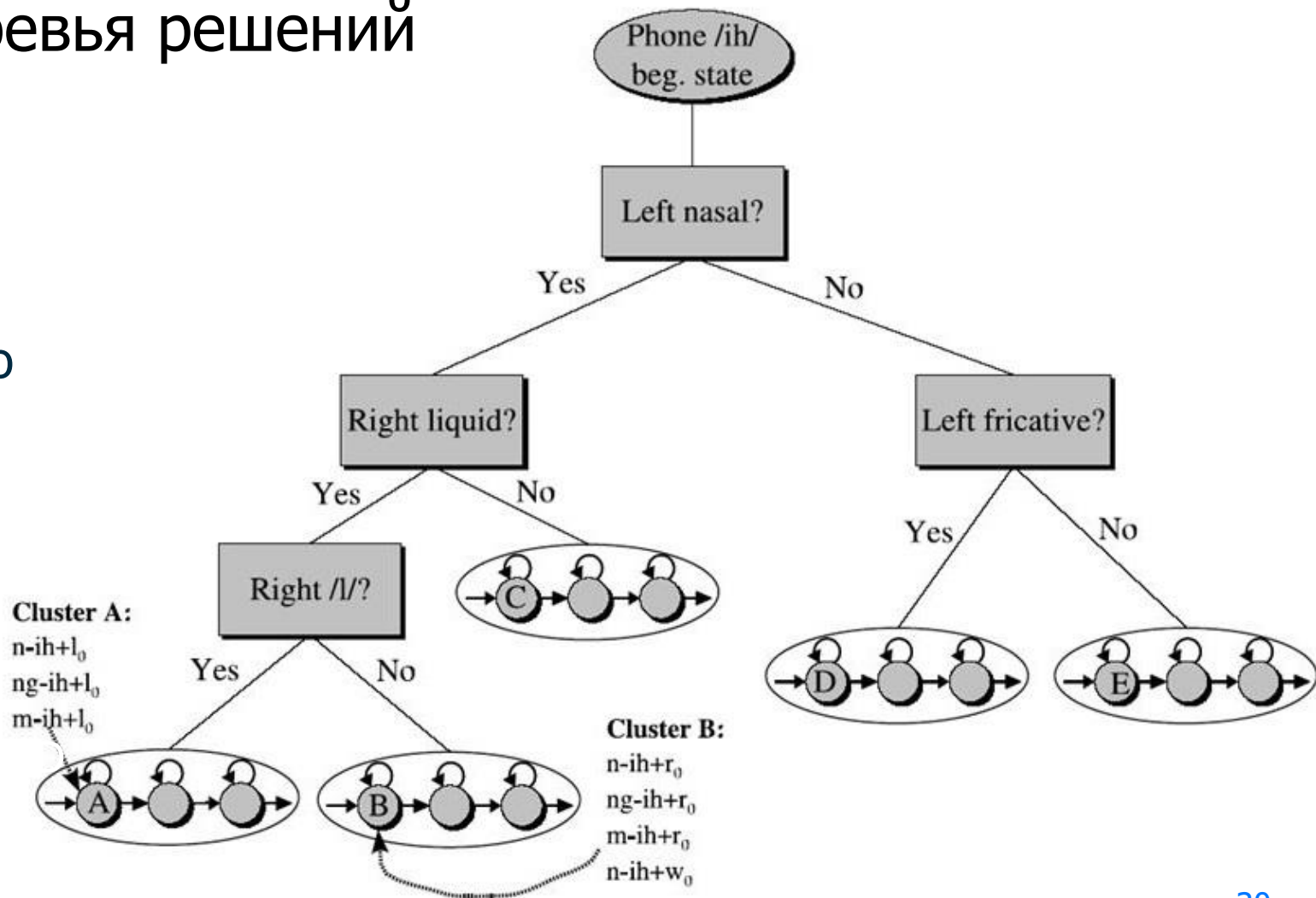
Tied triphones



Применение НММ для распознавания речи

Связывание состояний. Деревья решений

- Как правило, связывают трифоны, относящиеся к одной фонеме.
- Связывание обычно проводят по **дереву решений (decision tree)**.
- Дерево строится путем разбиения всего множества трифонов на классы в соответствие с «вопросами».
- Число связанных состояний — обычно 5-10 тысяч.



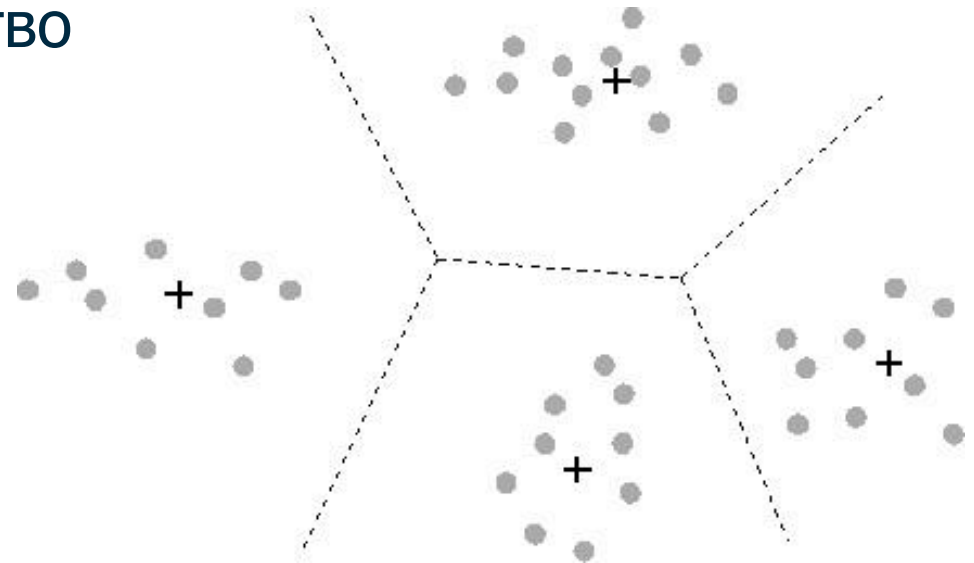
План лекции

- Скрытые Марковские Модели и связанные с ними задачи
- Применение НММ для распознавания речи
- Гауссовы смеси, обучение GMM-НММ
- Графы распознавания. Взвешенные конечные преобразователи. WFST-декодер. Словные сети
- Дискриминативное обучение GMM-НММ
- Адаптация систем распознавания речи

Гауссова смесь (Gaussian Mixture Model, GMM)

Непрерывные распределения в состояниях НММ:

- До сих пор мы рассматривали только дискретные распределения в состояниях
- Но часто наблюдения распределены непрерывно (например, MFCC-признаки)
- Можно провести векторное квантование и свести задачу к дискретной, но это «сжатие с потерями» и оно снижает качество
- Вероятности наблюдений в состояниях заменяются на плотности распределения: $b_j(o_t) = p(o_t | q_t = S_j)$
- Как использовать непрерывные распределения на практике?



Гауссова смесь (Gaussian Mixture Model, GMM)

- Многомерное нормальное (гауссово) распределение:

$$\mathcal{N}(o; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d |\det \Sigma|}} \exp \left\{ -\frac{1}{2} (o - \mu)^T \Sigma^{-1} (o - \mu) \right\}$$

- Гауссова смесь:

$$b_j(o_t) = \sum_{k=1}^M w_{jk} \mathcal{N}(o_t; \mu_{jk}, \Sigma_{jk}), \quad \text{где} \quad w_{jk} \geq 0, \quad \sum_{k=1}^M w_{jk} = 1.$$

- GMM – **генеративная** модель. Генерировать данные из нее очень просто:
 - Сначала генерируется номер компонента в соответствии с распределением $\{w_{jk}\}$
 - После этого генерируется вектор из распределения $\mathcal{N}(o_t; \mu_{jk}, \Sigma_{jk})$
- С помощью GMM с достаточно большим числом компонентов можно приблизить **любое непрерывное распределение** с достаточной точностью

Гауссова смесь (Gaussian Mixture Model, GMM)

- Пусть имеется набор данных o_t . Как описать его распределение гауссовой смесью?

$$p(o_t) = \sum_{k=1}^M w_k \mathcal{N}(o_t; \mu_k, \Sigma_k)$$

- Для обучения используется **метод максимального правдоподобия**
- Задача не решается в явном виде, применяется итерационный **ЕМ-алгоритм**.
- На Е-шаге вычисляются апостериорные вероятности компонентов:

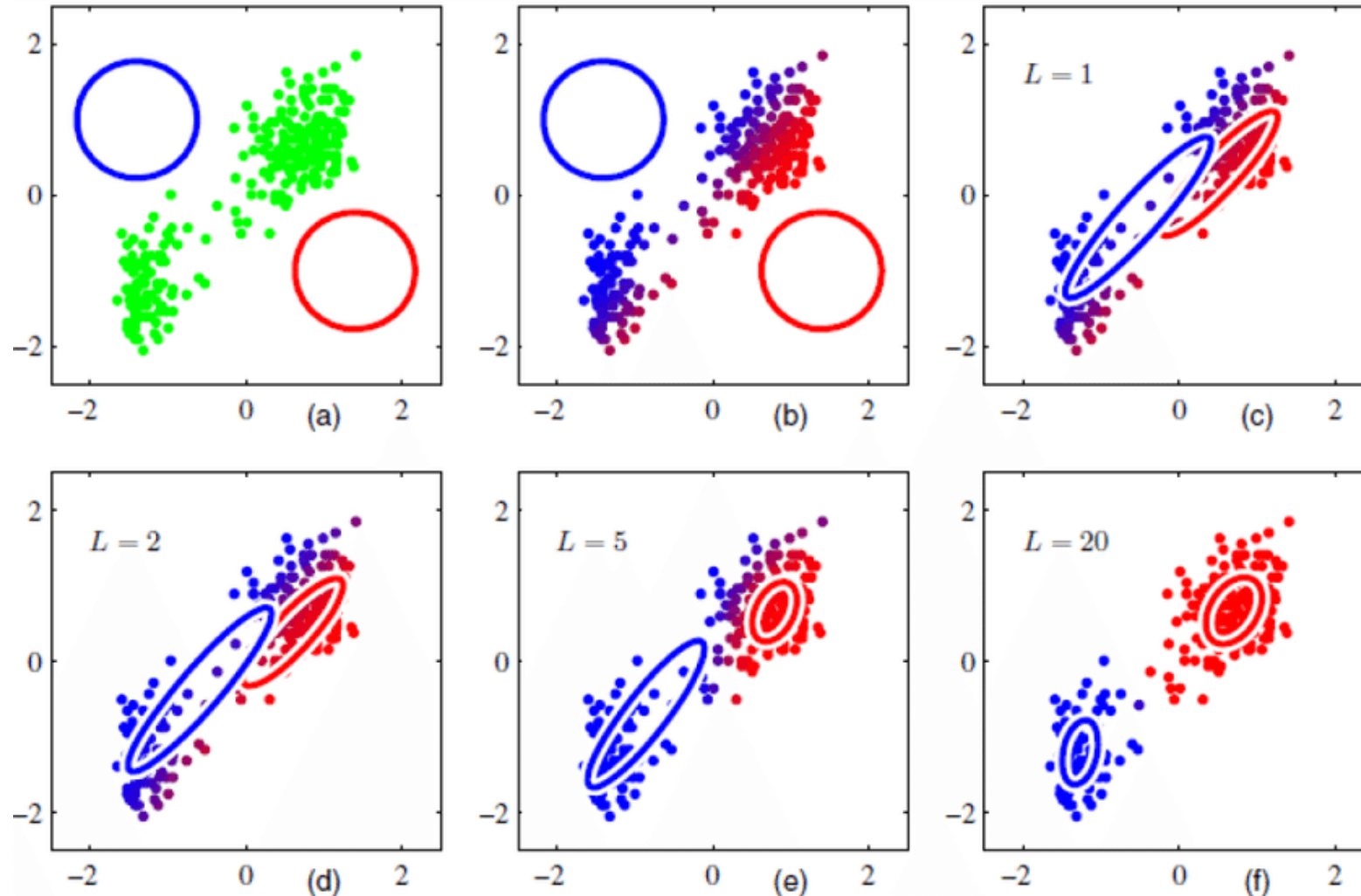
$$\gamma_{kt} = p(k|o_t) = \frac{w_k \mathcal{N}(o_t; \mu_k, \Sigma_k)}{\sum_{k=1}^M w_k \mathcal{N}(o_t; \mu_k, \Sigma_k)}$$

- На М-шаге вычисляются обновленные значения параметров:

$$\hat{w}_k = \frac{1}{T} \sum_{t=1}^T \gamma_{kt}, \quad \hat{\mu}_k = \frac{\sum_{t=1}^T \gamma_{kt} o_t}{\sum_{t=1}^T \gamma_{kt}}, \quad \hat{\Sigma}_k = \frac{\sum_{t=1}^T \gamma_{kt} (o_t - \hat{\mu}_k)(o_t - \hat{\mu}_k)^T}{\sum_{t=1}^T \gamma_{kt}}$$

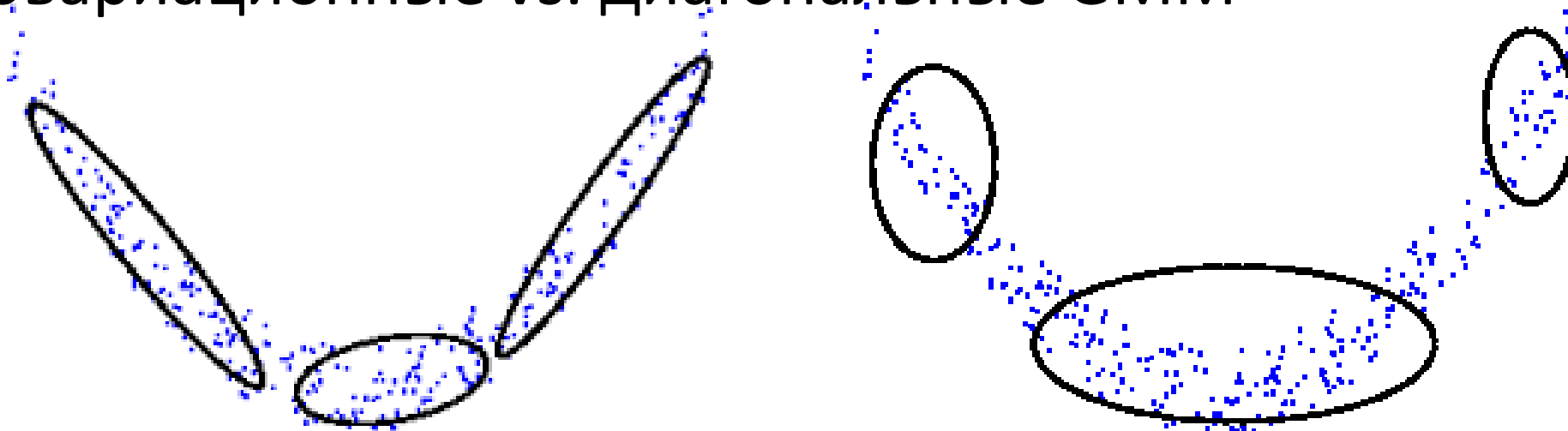
Гауссова смесь (Gaussian Mixture Model, GMM)

Иллюстрация работы EM-алгоритма



Гауссова смесь (Gaussian Mixture Model, GMM)

Полноковариационные vs. диагональные GMM



- GMM с **полноковариационными** матрицами – очень много параметров.
- GMM с **диагональными** матрицами – требуется больше компонент для аппроксимации
- Выход: **декорреляция** признаков
 - Использование MFCC (приблизительно декоррелированы благодаря DCT)
 - Использование PCA (Principal Component Analysis), оно же KLT (Karhunen-Loeve Transform) или выбеливание (whitening)

Скрытые Марковские модели с GMM в состояниях (GMM-HMM)

Алгоритм Баума-Уэлша для обучения GMM-HMM

- Е-шаг точно такой же, как для дискретных распределений в состояниях
- На М-шаге начальные вероятности и матрица переходов обновляются так же
- Для обновления параметров GMM можно вывести следующие соотношения:

$$\hat{w}_{jk} = \frac{\sum_t \gamma_{jk}(t)}{\sum_t \gamma_j(t)}, \quad \hat{\mu}_{jk} = \frac{\sum_t \gamma_{jk}(t) o_t}{\sum_t \gamma_{jk}(t)}, \quad \hat{\Sigma}_{jk} = \frac{\sum_t \gamma_{jk}(t) (o_t - \hat{\mu}_{jk})(o_t - \hat{\mu}_{jk})^T}{\sum_t \gamma_{jk}(t)}$$

где

$$\gamma_{jk}(t) = \gamma_j(t) \frac{w_{jk} \mathcal{N}(o_t; \mu_{jk}, \Sigma_{jk})}{b_j(o_t)}$$

называются вероятностями «посещения» k -го компонента смеси (**gaussian occupancy probabilities**) в состоянии j в момент времени t .

Скрытые Марковские модели с GMM в состояниях (GMM-HMM)

Достоинства и недостатки GMM-HMM

- + Возможность описывать практически любые распределения в состояниях
- + Эффективная процедура обучения
- Динамика ограничена марковским свойством
- Вероятности переходов не зависят от времени
- HMM предполагает, что наблюдения на соседних кадрах независимы и зависят только от состояния, в котором находится модель (frame independence assumption)
- GMM является «локальной» моделью и учится по ML-критерию
- Для повышения точности надо увеличивать количество компонент GMM, число параметров растет, модель переобучается, расчеты замедляются.

Скрытые Марковские модели с GMM в состояниях (GMM-HMM)

Независимость наблюдений – как «обойти» ?

- Использование дельта-признаков:
 - Пусть имеется последовательность векторов $O = (o_1, o_2, \dots, o_T)$
 - Дополним векторы наблюдений «производными»:

$$\Delta o_t = (o_{t+1} - o_{t-1})/2 \approx \partial o_t / \partial t$$

$$\Delta \Delta o_t = o_{t+1} - 2o_t + o_{t-1} \approx \partial^2 o_t / \partial^2 t$$

- Frame stacking (splicing): объединение векторов признаков вокруг текущего в один длинный «супервектор»:

$$O_t = [o_{t-l}^T, \dots, o_{t-1}^T, o_t^T, o_{t+1}^T, \dots, o_{t+r}^T]^T$$

Скрытые Марковские модели с GMM в состояниях (GMM-НММ)

Генеративность и локальность GMM:

- **Локальность:** пространство оказывается разделено на «области» в каждой из которых ДОМИНИРУЕТ только один компонент GMM, а остальные не оказывают существенного влияния на значения правдоподобия.
- **Генеративность:** параметры генеративных моделей, как правило, выбираются согласно критерию максимального правдоподобия. Т.е. так, чтобы максимизировать правдоподобие ВЕРНОЙ гипотезы. Но если «конкурирующие» гипотезы близко, то качество распознавания будет низким.
- **Дискриминативное обучение:** идея – максимально отделить правильную гипотезу от всех конкурирующих. Подробнее – в конце лекции.

Скрытые Марковские модели с GMM в состояниях (GMM-HMM)

Переобучение и усложнение GMM – как бороться?

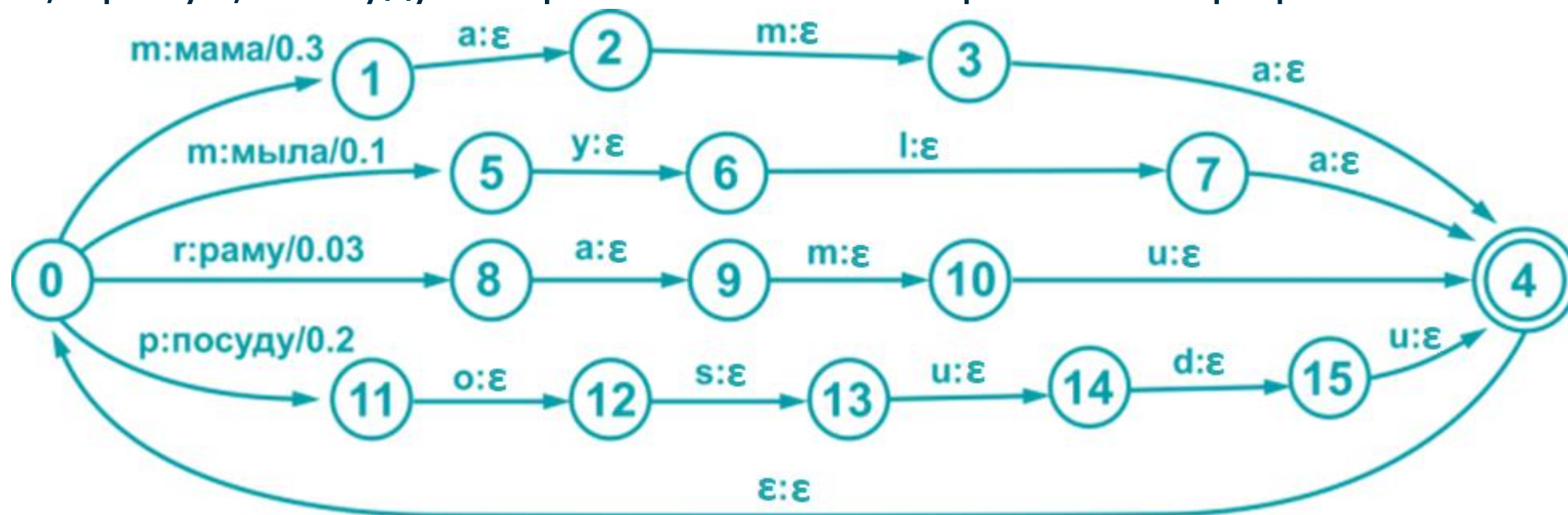
- **Связывание** параметров (не только состояний, но и параметров GMM, отдельных гауссиан, весов, вероятностей переходов)
- **Векторное квантование** признаков и использование **дискретных** распределений
- Создание большого **пула** гауссиан и «набор» отдельных GMM в состояниях из этого пула с различными весами. На этой идее основаны SGMM (subspace GMM)
- **Общий вывод:** GMM – не самый лучший из возможных классификаторов в состояниях HMM...

План лекции

- Скрытые Марковские Модели и связанные с ними задачи
- Применение НММ для распознавания речи
- Гауссовы смеси, обучение GMM-НММ
- Графы распознавания. Взвешенные конечные преобразователи. WFST-декодер. Словные сети
- Дискриминативное обучение GMM-НММ
- Адаптация систем распознавания речи

Распознавание с помощью суб-словных моделей

- Пусть требуется распознавать всевозможные последовательности из слов «мама», «мыла», «раму», «посуду». Строим из этих слов фонемный граф:



- Этот граф можно преобразовать в трифонный и, далее, в «стейтовый»
- На стейтовом графе можно искать лучший (Витерби) путь как в одной большой НММ. Это можно делать с помощью [token-passing](#) алгоритма!

Распознавание с помощью суб-словных моделей

- **Гипотеза** – тот же токен из token-passing алгоритма. Хранит в себе пройденный путь (чтобы можно было восстановить последовательность слов) и накопленный логарифм правдоподобия
- На каждом новом кадре
 - Вычисляются вероятности всех состояний всех НММ
 - Все активные гипотезы расширяются с учетом возможных переходов из состояния, правдоподобий в состояниях и вероятностей переходов
 - В каждом состоянии запоминается только лучшая гипотеза (**алгоритм Витерби**)
- Если граф большой, то число активных гипотез растет очень быстро. Выход – отсекать (prune) «малоперспективные» гипотезы (beam pruning, histogram pruning,...)

Декодирование с языковой моделью

- Простейший сценарий:
 - В гипотезах (токенах) хранится пройденный путь.
 - Как только дошли до конца очередного слова запрашиваем у ЯМ его вероятность при данной истории и добавляем ее логарифм в score гипотезы
- Недостатки простейшего сценария:
 - Вероятность гипотезы меняется «скачкообразно»
 - Гипотеза может «выпасть» из beam'a до того, как ее score улучшится благодаря ЯМ
 - Частые обращения к ЯМ, дублирование запросов на похожих гипотезах
- Возможные решения:
 - «Внедрить» языковые вероятности непосредственно в стейтовый граф
 - «Размазать» их по длине слова

Построение графа распознавания

- Что надо иметь для построения стейтового графа:
 - Языковая модель или грамматика – показывает возможные переходы из слова в слово (с их вероятностями)
 - Лексикон (словарь транскрипций) – показывает, как произносятся слова, т.е. из каких фонем оно состоит. Может быть несколько транскрипций на слово, причем с разными вероятностями
 - Контекстная информация – какие трифоны получаются из фонем с учетом левого/правого контекстов и связывания состояний
 - Акустическая модель (НММ) – показывает из каких состояний состоит каждый трифон и задает вероятности переходов из состояния в состояние
- К счастью, каждый из этих видов информации можно представить в едином формате – **Weighted Finite-State Transducer** (WFST). WSFT является **вероятностным конечным автоматом**, трансформирующим входную последовательность символов в выходную.

Взвешенные конечные преобразователи (WFST)

- **Конечный автомат (Finite State Machine, FSM)**: система, которая может находиться в конечном числе состояний и переходить из одного в другое при получении тех или иных входных данных:
 - **Множество состояний** Q , в нем выделяются **начальные** состояния $I \subset Q$ и **конечные** состояния $F \subset Q$;
 - **Входной алфавит** Σ , последовательность элементов которого поступает на вход;
 - **Функция перехода** $\delta(q, a)$ определяет куда перейдет система, находящаяся в состоянии $q \in Q$, получив на вход токен $a \in \Sigma$ (либо специальный «пустой» символ ε)
 - В начале работы система находится в одном из состояний $q_0 \in I$.
 - На каждом шаге система считывает один токен $a \in \Sigma$ и делает переход согласно функции $\delta(q, a)$
 - Если по окончании входной последовательности система находится в состоянии $q \in F$, то говорят, что система **принимает** данную последовательность.
- Конечный автомат, направленный лишь на определение того, принимается ли данная последовательность, называются **акцепторами (Finite State Acceptors, FSA)**

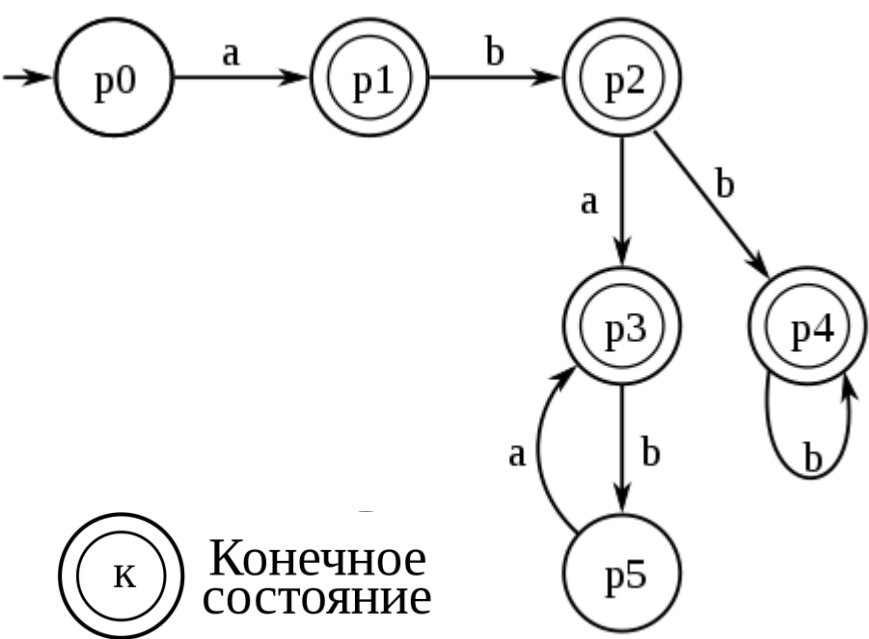
Взвешенные конечные преобразователи (WFST)

- Примеры задания конечного автомата

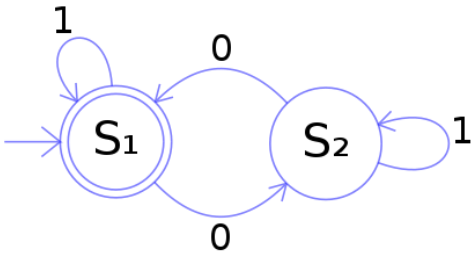
- Таблица переходов:

| Исходное состояние | Следующее состояние | | |
|--------------------|---------------------|------------------|---------------------|
| | Входной символ а | Входной символ b | Любой другой символ |
| p0 | p1 | p0 | p0 |
| p1 | p1 | p2 | p1 |
| p2 | p3 | p4 | p2 |
| p3 | p3 | p5 | p3 |
| p4 | p4 | p4 | p4 |
| p5 | p3 | p5 | p5 |

- Граф переходов:



- Пример акцептора, принимающего строки из нулей и единиц с ЧЁТНЫМ количеством нулей:



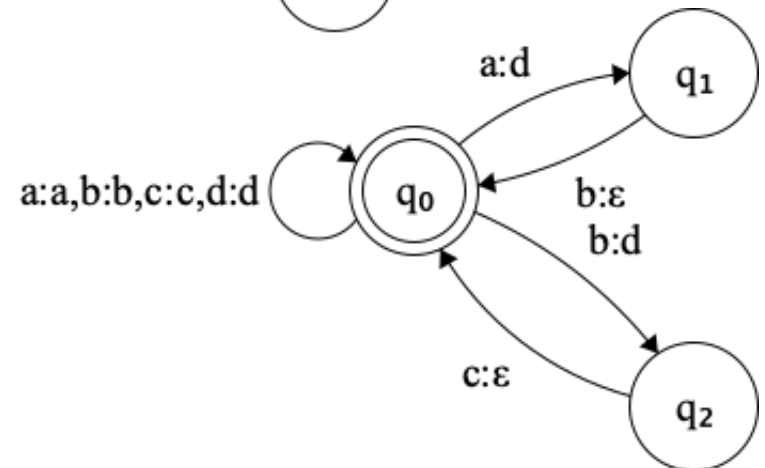
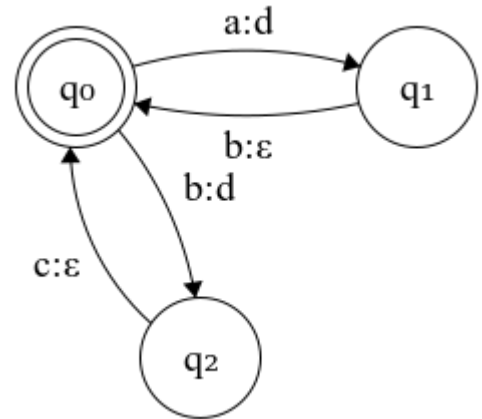
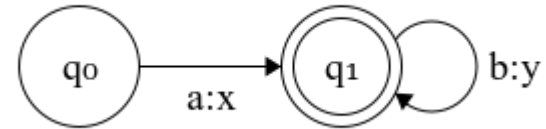
Взвешенные конечные преобразователи (WFST)

- **Конечный преобразователь (Finite State Transducer, FST)**: аналог FSM, который не просто принимает входную последовательность, но и генерирует выходную.
 - На каждом шаге система считывает один токен $a \in \Sigma \cup \{\varepsilon\}$ и не только делает переход согласно функции $\delta(q, a)$, но и выдает выходной токен $b \in \Gamma \cup \{\varepsilon\}$, где Γ – **выходной** алфавит
 - FST удобно представлять в виде **ориентированного графа**, в котором узлы соответствуют состояниям, а на ребрах которого заданы пары из входного и выходного токенов $a: b$.
 - Говорят, что преобразователь T **преобразует** последовательность x в последовательность y (обозначение $x[T]y$) если существует путь по этому графу из какого-то **начального** состояния в какое-то **конечное**, на котором последовательность входных токенов равна x , а последовательность выходных токенов равна y .
- **Операции над FST**:
 - **Объединение** $T \cup S$: объединяются множества путей в двух графах
 - **Конкатенация** $T \cdot S$: все пути второго графа приклеиваются к путям первого
 - **Композиция** $T \circ S$: $x[T \circ S]y$ тогда и только тогда, когда существует такая последовательность z , что $x[T]z$ и $z[S]y$. Т.е. композиция как бы последовательно применяет преобразования сначала от T , а потом от S .

Взвешенные конечные преобразователи (WFST)

- Примеры конечных преобразователей

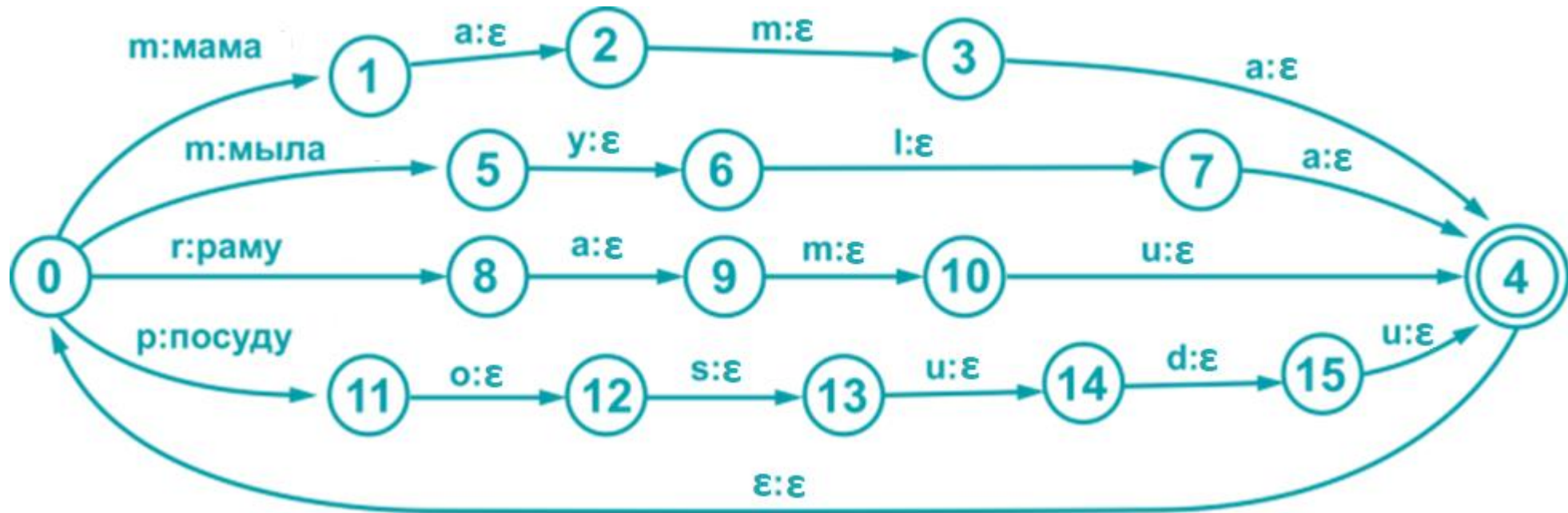
- Принимает строки вида «a», «ab», «abb» и т.д. и преобразует их в «x», «xu», «хуу» и т.д.:
- Принимает последовательности из пар «ab» или «bc» и преобразует их в соответствующее количество букв «d»:
- Принимает любые строки, состоящие из букв «a», «b», «c», «d» и может в них опционально заменять пары «ab» или «bc» на «d»



Взвешенные конечные преобразователи (WFST)

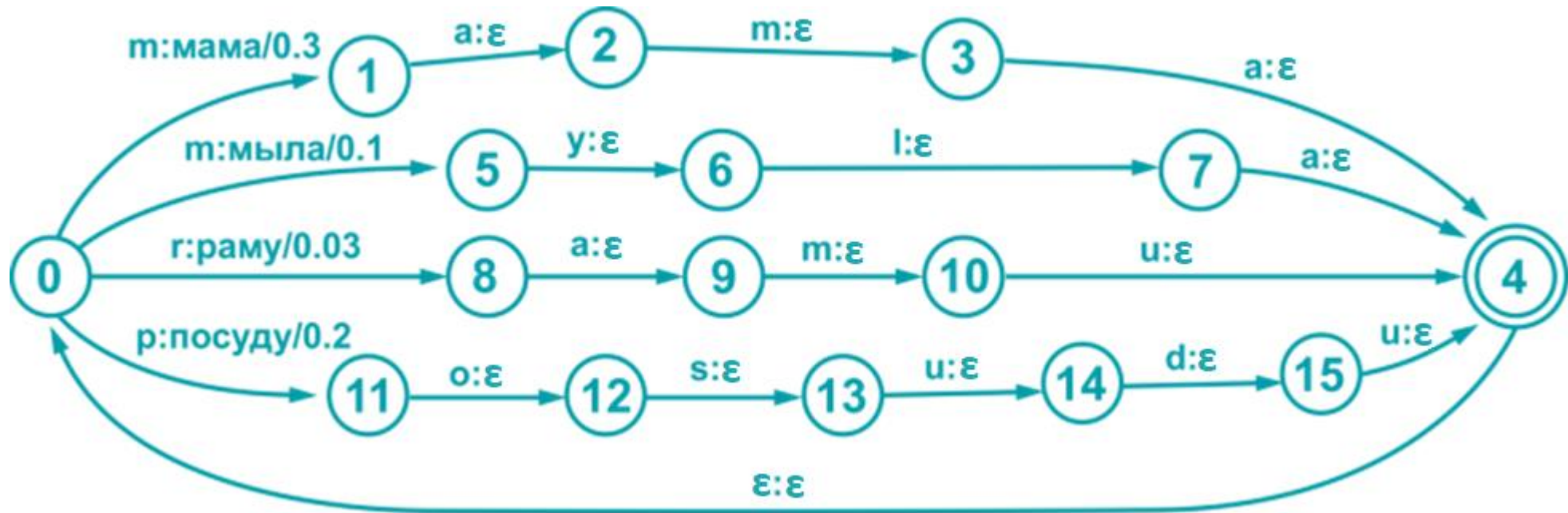
- Примеры конечных преобразователей

- Принимает последовательности фонем, соответствующих словам «мама», «мыла», «раму», «посуду» в любом порядке и количестве, и преобразует их в последовательность этих слов:



Взвешенные конечные преобразователи (WFST)

- Взвешенный конечный преобразователь (Weighted Finite State Transducer, WFST): каждому ребру, а также начальным и конечным узлам графа сопоставлен **БЕС**



Взвешенные конечные преобразователи (WFST)

- Взвешенный конечный преобразователь (Weighted Finite State Transducer, WFST): каждому ребру, а также начальным и конечным узлам графа сопоставлен **ВЕС**
 - Все веса берутся из некоторого **полукольца** \mathbb{K} , т.е. их можно «**складывать**» \oplus и «**умножать**» \otimes
 - Вес пути** π , состоящего из ребер e_1, e_2, \dots, e_k , равен «произведению» весов ребер: $w[\pi] = w[e_1] \otimes \dots \otimes w[e_k]$
 - Пусть $P(x, y)$ – множество путей π , из какого-то начального узла в какой-то конечный узел с входной последовательностью x и выходной последовательностью y , а $i(\pi)$ и $f(\pi)$ – это начальный и конечный узлы пути π .
 - Тогда **вес преобразования** $x[T]y$ определяется как «сумма» по всем путям из $P(x, y)$:

$$\llbracket T \rrbracket(x, y) = \bigoplus_{\pi \in P(x, y)} w[i(\pi)] \otimes w[\pi] w[f(\pi)].$$

- Если веса – это **вероятности**, то $\llbracket T \rrbracket(x, y)$ – суммарная вероятность всех путей из $P(x, y)$. В этом случае \mathbb{K} – **вероятностное полукольцо**, а \oplus и \otimes – обычные арифметические сложение и умножение
- Если веса – это **логарифмы вероятностей**, то \mathbb{K} – **логарифмическое полукольцо**: в этом случае $a \otimes b = a + b$ и $a \oplus b = \log(e^a + e^b)$
- Различные преобразования/пути можно сравнивать по их весу и выбирать **оптимальные**

Построение WFST-графа распознавания

- Традиционно, WFST для отдельных компонентов обозначают так:
 - G – для **грамматики** или n -граммной ЯМ (переводит слова в предложение)
 - L – для **лексикона** (переводит фонемы в слова)
 - C – для **контекстной** зависимости (переводит трифоны в фонемы)
 - H – для **HMM** (переводит состояния трифонов в сами трифоны)
- С помощью **композиции** этих WFST можно построить преобразователь, который транслирует последовательность состояния фонем прямо в предложение!

$$W = H \circ C \circ L \circ G$$

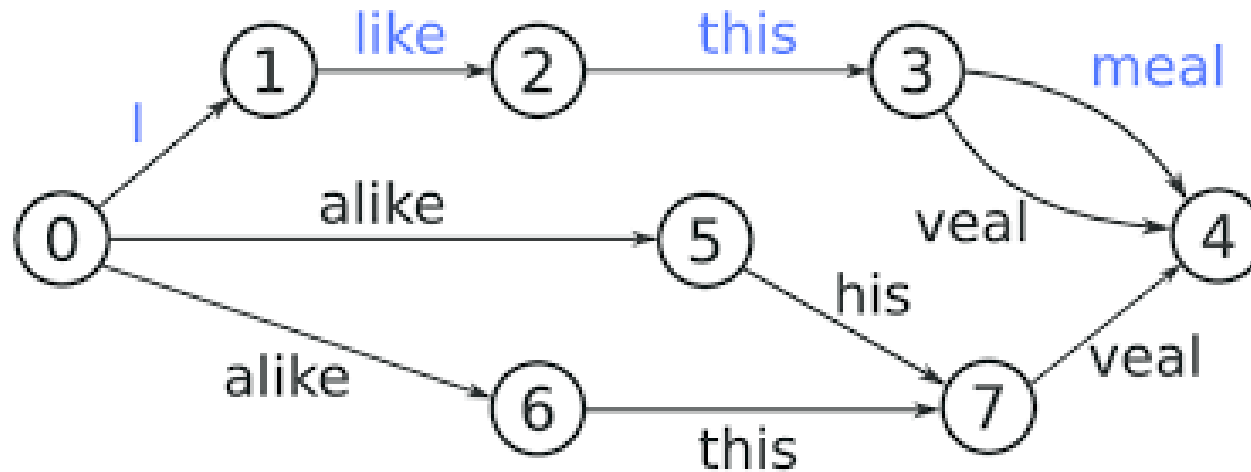
- Такой WFST фактически определяет **граф распознавания** для декодера.
- В ходе построения граф дополнительно оптимизируют с помощью операций **минимизации** и **детерминизации**, а также «проталкивают» **веса** ближе к началу
- OpenFST – open-source библиотека для работы с WFST

WFST-декодер: базовые параметры

- Для настройки функционирования системы ASR есть ряд параметров:
 - **Ширина луча поиска**, beam width (для beam pruning)
 - **Максимально допустимое количество гипотез** (токенов), maximum hypotheses number (для histogram pruning)
 - **Штраф за вход в слово** α (word insertion penalty). Заставляет декодер предпочитать более длинные слова коротким.
 - **Вес языковой модели** β (lm scale).
- С учетом α и β декодер ищет последовательность слов W , которая максимизирует следующую величину:
$$\arg \max_W [\log p(O|W) + \alpha |W| + \beta \log P(W)] = \arg \max_W p(O|W) |W|^\alpha P(W)^\beta$$
- С помощью настройки этих параметров можно регулировать соотношение между точностью и скоростью работы системы распознавания

WFST-декодер: N-best списки и словные сети, рескоринг

- Иногда требуется вернуть не только лучшую гипотезу, а N лучших.
- Для этого в состояниях надо хранить не по одному, а по несколько лучших токенов!
- Список лучших гипотез называют **N-best list**.
- Более компактное представление набора лучших гипотез – **словная сеть (word lattice)**:



- Часто правильная гипотеза – не лучшая в списке/сети из-за «слабой» языковой модели
- В этом случае можно применить **рескоринг (re-scoring)**: пересчитать веса гипотез с помощью продвинутой ЯМ (n-граммной с увеличенным n , нейронной)

План лекции

- Скрытые Марковские Модели и связанные с ними задачи
- Применение НММ для распознавания речи
- Гауссовы смеси, обучение GMM-НММ
- Графы распознавания. Взвешенные конечные преобразователи. WFST-декодер. Словные сети
- **Дискриминативное обучение GMM-НММ**
- Адаптация систем распознавания речи

Дискриминативное обучение GMM-НММ

- Традиционный метод обучения GMM-НММ – метод максимального правдоподобия:

$$\hat{\lambda} = \operatorname{argmax}_{\lambda} P(O|W, \lambda)$$

- НО: он «тянет вверх» правдоподобие ТОЛЬКО на истинных последовательностях слов!
- При этом правдоподобие «альтернативных» гипотез может быть очень близко
- Дискриминативные критерии стремятся максимально отделить (discriminate) истинную гипотезу от всех остальных.
- Максимум взаимной информации (MMI):

$$\hat{\lambda} = \operatorname{argmax}_{\lambda} P(W|O, \lambda) = \operatorname{argmax}_{\lambda} \frac{P(O|W, \lambda)P(W)}{\sum_{W'} P(O|W', \lambda)P(W')}$$

- Суммирование в знаменателе проводится по всем альтернативным гипотезам (lattice!)
- Обучение: расширенный алгоритм Баума-Уэлша (extended BW, EBW).
- Другие критерии: Minimum Word/Phone Error (MWE/MPE), Boosted MMI (BMMI)

Расширенный алгоритм Баума-Уэлша (extended BW, EBW)

- Базовый алгоритм Баума-Уэлша:

$$\hat{\mu}_{jk} = \frac{\sum_t \gamma_{jk}(t) o_t}{\sum_t \gamma_{jk}(t)}, \quad \hat{\Sigma}_{jk} = \frac{\sum_t \gamma_{jk}(t) (o_t - \hat{\mu}_{jk})(o_t - \hat{\mu}_{jk})^T}{\sum_t \gamma_{jk}(t)}$$

- Для всех дискриминативных критериев EBW имеет одинаковую форму

$$\hat{\mu}_{jk} = \frac{\sum_t \left(\gamma_{jk}^{NUM}(t) - \gamma_{jk}^{DEN}(t) \right) o_t}{\sum_t \left(\gamma_{jk}^{NUM}(t) - \gamma_{jk}^{DEN}(t) \right)}$$
$$\hat{\Sigma}_{jk} = \frac{\sum_t \left(\gamma_{jk}^{NUM}(t) - \gamma_{jk}^{DEN}(t) \right) (o_t - \hat{\mu}_{jk})(o_t - \hat{\mu}_{jk})^T}{\sum_t \left(\gamma_{jk}^{NUM}(t) - \gamma_{jk}^{DEN}(t) \right)}$$

- В них $\gamma_{jk}^{NUM}(t)$, $\gamma_{jk}^{DEN}(t)$ - вероятности «посещения» гауссовых компонент, вычисленные по сетям (lattice) числителя и знаменателя.
- Для разных критериев немного отличаются лишь алгоритмы оценки $\gamma_{jk}^{NUM}(t)$ и $\gamma_{jk}^{DEN}(t)$

План лекции

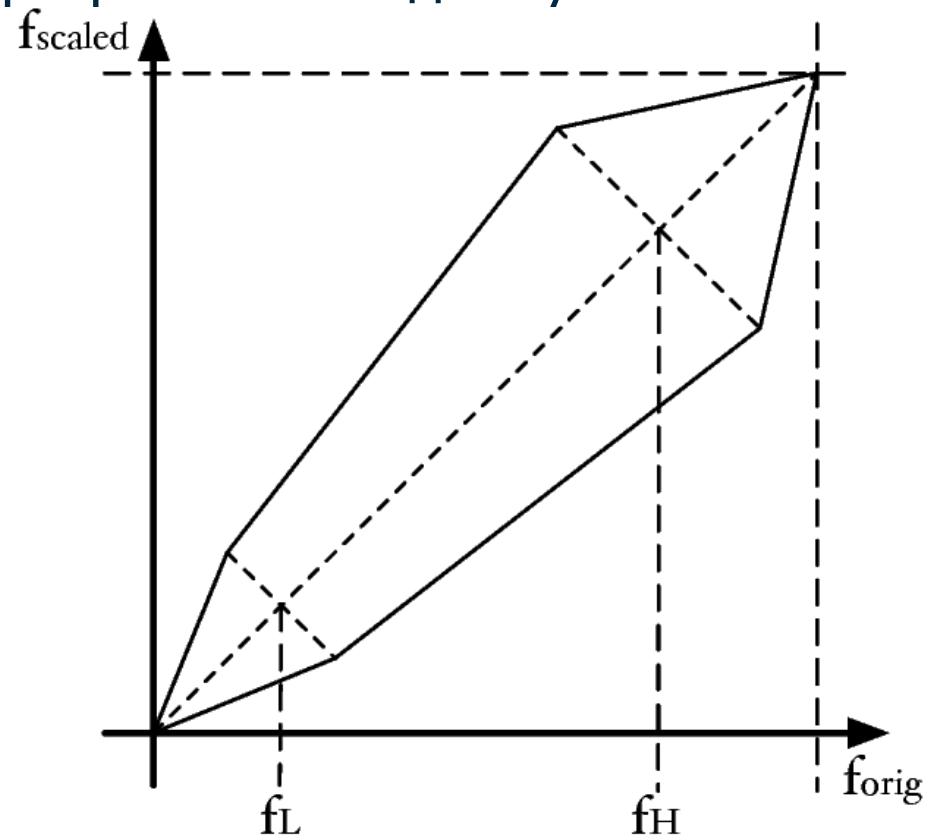
- Скрытые Марковские Модели и связанные с ними задачи
- Применение НММ для распознавания речи
- Гауссовы смеси, обучение GMM-НММ
- Графы распознавания. Взвешенные конечные преобразователи. WFST-декодер. Словные сети
- Дискриминативное обучение GMM-НММ
- Адаптация систем распознавания речи

Адаптация систем распознавания речи

- Современные системы, как правило, **дикторонезависимые (Speaker Independent, SI)**
- Система, настроенная на конкретный голос, – **дикторозависимая (Speaker Dependent, SD)**
- **Адаптация** – способ улучшить распознавание конкретного голоса дикторонезависимой системой по небольшому набору данных для этого голоса
- Типы адаптации:
 - **Контролируемая (supervised)** – даны фонограммы голоса и их текстовые расшифровки
 - **Неконтролируемая (unsupervised)** – даны только фонограммы
 - **Оффлайн** – фонограммы записываются, все разом обрабатываются, на выходе адаптированная модель
 - **Онлайн** – постепенная адаптация по мере поступления данных целевого голоса
- Для пользователя удобнее всего неконтролируемая онлайн адаптация (от него вообще ничего не требуется). Но она наиболее сложна в реализации и работает не так хорошо
- **Дикторо-адаптивное обучение (Speaker Adaptive Training, SAT)**: в ходе обучения применяется такая же «адаптация», как и при распознавании.

Нормализация длины голосового тракта

- В зависимости от длины голосового тракта меняется основной тон (ОТ) речи
- Чем выше основной тон, тем спектрограмма более «растянута» по частоте
- Идея для адаптации: растянуть/сжать все спектрограммы к ~одному значению ОТ до вычисления mel-fbanks
- Такой способ называется **Vocal Tract Length Normalization (VTLN)**
- На большей части частотного диапазона преобразование линейное
- Как искать коэффициент растяжения/сжатия?
- Перебор значений 0.9:0.02:1.1 и выбор того, на котором максимально правдоподобие SI-модели на адаптационных данных



Maximum Likelihood Linear Regression (MLLR)

- Пусть имеется построенная GMM-HMM, ее правдоподобие максимально на обучающих данных. Но данные нового диктора распределены немного иначе...
- Идея: давайте для каждого компонента каждой GMM немного «подправим» средние векторы, чтобы **максимизировать правдоподобие** на новых данных
- Как подправим? **Аффинным преобразованием**: $\mu_{jk}^{new} = A_{jk}\mu_{jk} + b_{jk}$
- Решение итерационное, **ЕМ-алгоритм**
- Можно адаптировать не только средние, но и ковариации
- Если гауссовых компонентов много, а данных мало – переобучение. Поэтому для «похожих» гауссиан ищут общее преобразование. Гауссианы «кластеризуют» с помощью дерева, чем больше данных, тем больше кластеров можно преобразовывать.
- MLLR хорошо работает даже на очень небольших объемах данных (десятки секунд)

Constrained MLLR (CMLLR) / feature-space MLLR (fMLLR)

- В MLLR мы двигали компоненты GMM к данным
- Но можно же и в обратную сторону!
- Идея: давайте найдем такое аффинное преобразование координат, чтобы **максимизировать правдоподобие** на преобразованных новых данных:

$$o_t^{new} = A o_t + b$$

- Решение опять итерационное, **ЕМ-алгоритм**
- Ищется только одна матрица и один вектор, значит шанс переобучиться минимален
- CMLLR/fMLLR хорошо подходит для дикторo-адаптивного обучения:
 - Обучим дикторoнезависимую модель
 - Для каждого диктора из обучающей выборки найдем преобразование
 - Переучим модель на преобразованных данных
 - **НО в test-time без преобразования уже ничего работать НЕ БУДЕТ!**

Maximum A Posteriori (MAP): байесовская адаптация

- В MLLR/fMLLR мы максимизировали правдоподобие модели на адаптационных данных
- В методе MAP ищутся параметры модели, максимизирующие апостериорное распределение, полученное в результате «наблюдения» адаптационных данных:

$$\hat{\lambda} = \operatorname{argmax}_{\lambda} p(\lambda | O^{adapt}) = \operatorname{argmax}_{\lambda} p(O^{adapt} | \lambda) p(\lambda)$$

- По сравнению с ML добавилось априорное распределение $p(\lambda)$ – это просто SI-модель
- Если бы его не было, получилась бы стандартная оценка для среднего:

$$\mu_{jk}^{SD} = \frac{\sum_t \gamma_{jk}(t) o_t}{\sum_t \gamma_{jk}(t)} = \frac{\sum_t \gamma_{jk}(t) o_t}{c_{jk}}$$

- А получается так:

$$\hat{\mu}_{jk} = \frac{\tau_{jk} \mu_{jk}^{SI} + \sum_t \gamma_{jk}(t) o_t}{\tau_{jk} + \sum_t \gamma_{jk}(t)} = \frac{\tau_{jk} \mu_{jk}^{SI} + c_{jk} \mu_{jk}^{SD}}{\tau_{jk} + c_{jk}}$$

- Получается, что компоненты, которые не «посещались», не изменятся

Сравнение MLLR и MAP

- MLLR хорошо работает на очень малых объемах адаптационных данных
- MLLR легко адаптировать к различным объемам доступных данных
- MAP на малых объемах адаптационных данных адаптируется плохо
- Но с ростом объема MAP улучшается и в конце концов обгоняет MLLR
- В пределе MAP-решение стремится к ML-решению, т.е. к чистой SD-модели
- Существуют различные комбинации этих подходов, которые сочетают достоинства обоих

Литература к этой лекции

- L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE, 77(2):257–286, 1989. – [про HMM и основные задачи](#)
- S. J. Young, J. J. Odell, and P. C. Woodland. Tree-based state tying for high accuracy acoustic modelling. In Proceedings of the Workshop on Human Language Technology, HLT'94, pages 307–312, 1994. – [про деревья решений и связывание состояний](#)
- Jeff A. Bilmes. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical Report ICSI-TR-97-021, University of Berkeley, 4(6), 2000. – [про EM-алгоритм для GMM и GMM-HMM](#)
- Mehryar Mohri. Weighted Finite-State Transducer Algorithms. An Overview, pages 551–563. Springer Berlin Heidelberg, 2004. – [про WFST](#)
- M. J. F. Gales. Maximum likelihood linear transformations for hmm-based speech recognition. Computer Speech & Language, 12(2):75–98, 1998. – [про MLLR](#)



Спасибо
за внимание!

Вопросы?