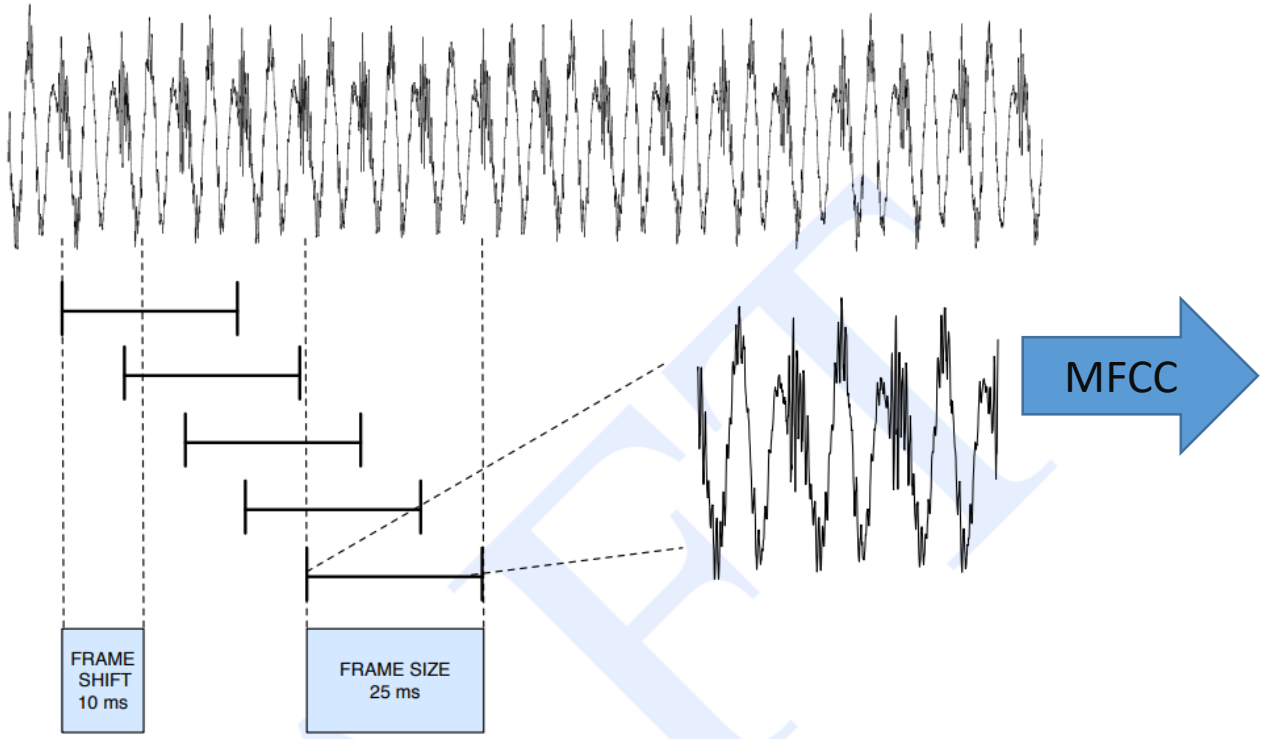
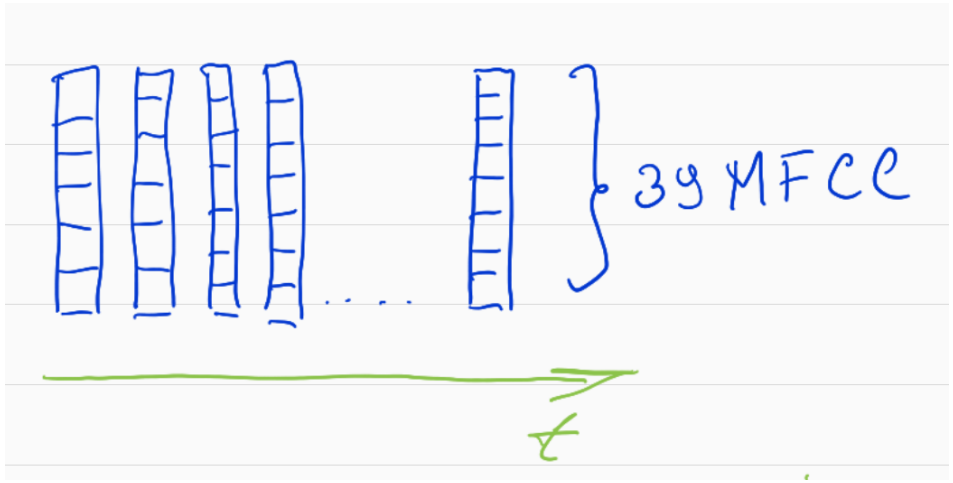
Распознавание речи

Классический подход

Извлечение признаков

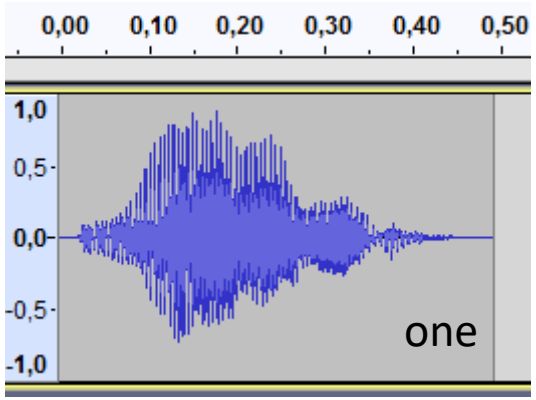


MFCC



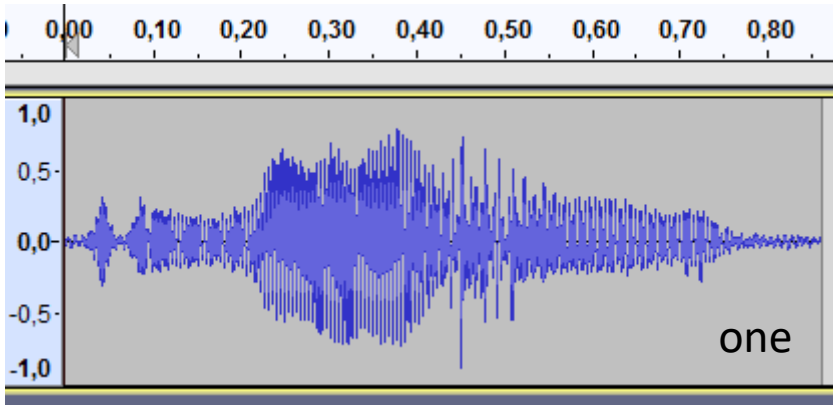
В итоге получаем количество признаков равное количеству окон на интервале времени t

Распознаем отдельные слова



фреймы

Референсное слово



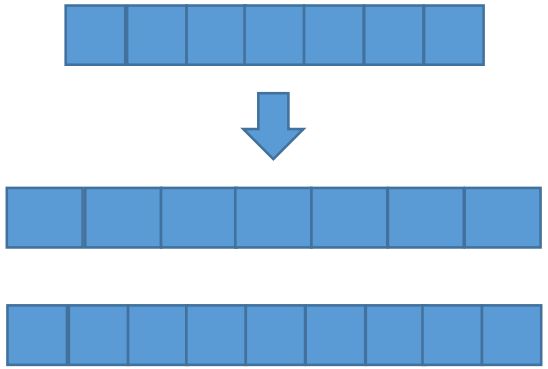
фреймы

Слово для распознавания

- Есть база шаблонных слов
- Нужно сопоставить два слова и сказать насколько слово для распознавания похоже на референсное слово
- Проблема длина последовательностей фреймов разная

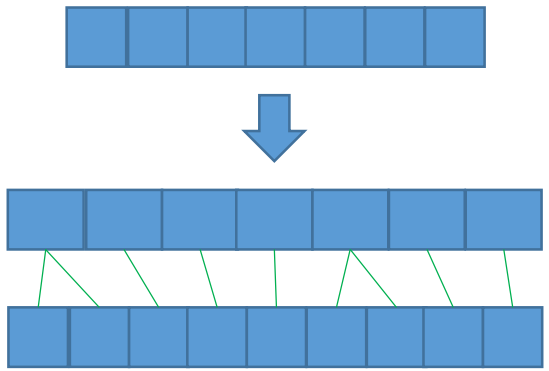
Линейное трансформация временной шкалы (Linear time warping)

- Нам нужны две операции для нахождения расстояния между двумя последовательностями (фреймов)
  - Найти выравнивание между последовательностями
  - Сложить локальные расстояния между выравненными последовательностями
- Простое выравнивание это растянуть линейно более короткую последовательность



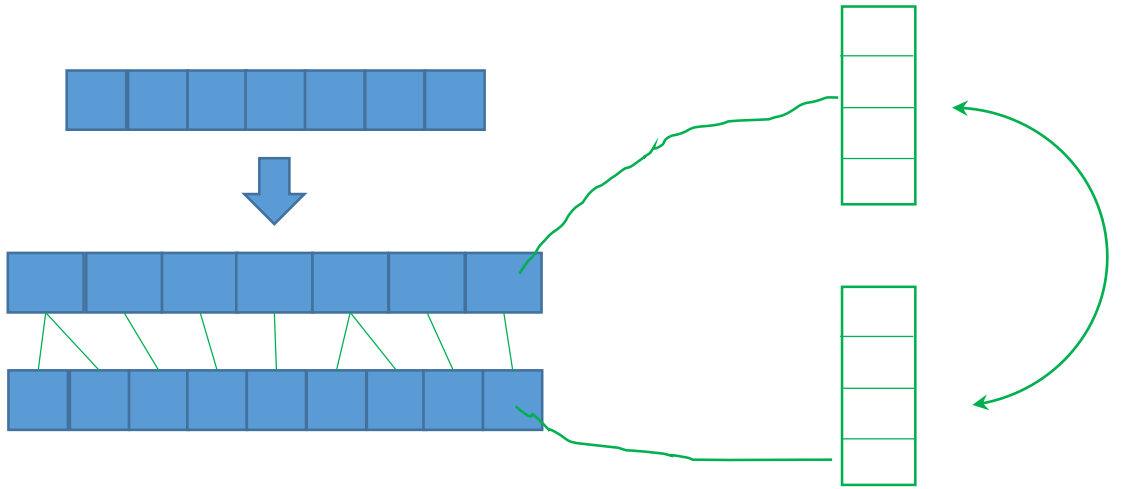
Линейное трансформация временной шкалы (Linear time warping)

- Нам нужны две операции для нахождения расстояния между двумя последовательностями (фреймов)
  - Найти выравнивание между последовательностями
  - Сложить локальные расстояния между выравненными последовательностями
- Простое выравнивание это растянуть линейно более короткую последовательность



Линейное трансформация временной шкалы (Linear time warping)

- Нам нужны две операции для нахождения расстояния между двумя последовательностями (фреймов)
  - Найти выравнивание между последовательностями
  - Сложить локальные расстояния между выравненными последовательностями
- Простое выравнивание это растянуть линейно более короткую последовательность



Расстояние между фреймами – это расстояние между векторами признаков MFCC

Оценка расстояния

- Имея две выравненные последовательности X и Y мы можем оценить расстояние между ними
  - $D(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{k=1}^{n}(x_k - y_k)}$ - расстояние между двумя окнами последовательностей
    - $\boldsymbol{x} = (x_1, x_2, \dots x_n)$ и $\boldsymbol{y} = (y_1, y_2, \dots, y_n)$ – вектор MFCC признаки одного фрейма последовательностей
  - Сумма расстояний выровненных фреймов – расстояние межу двумя последовательностями

Что не так с наивным подходом?

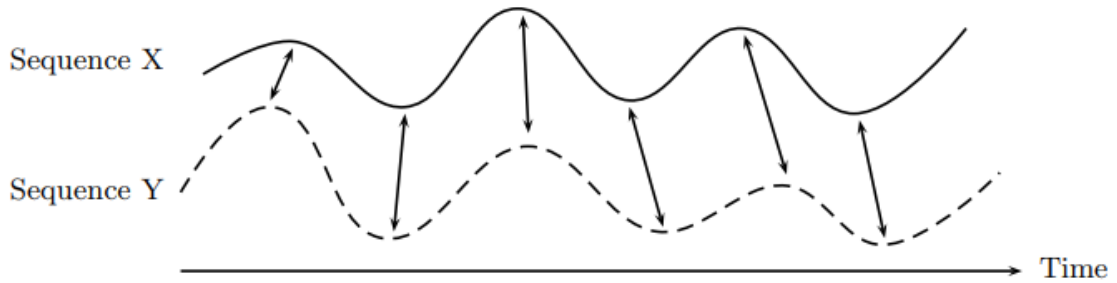
- Выравнивание через линейное растяжение для речи неприемлемо
  - У разных звуков разная длительность, и разная “способность к растяжению” (гласные, согласные)
- Оценка расстояния
  - Одинакова для всей последовательности (соседние фреймы могут перекрывать разные звуки)
- Нужен алгоритм автоматического выравнивания двух последовательностей равной длинны

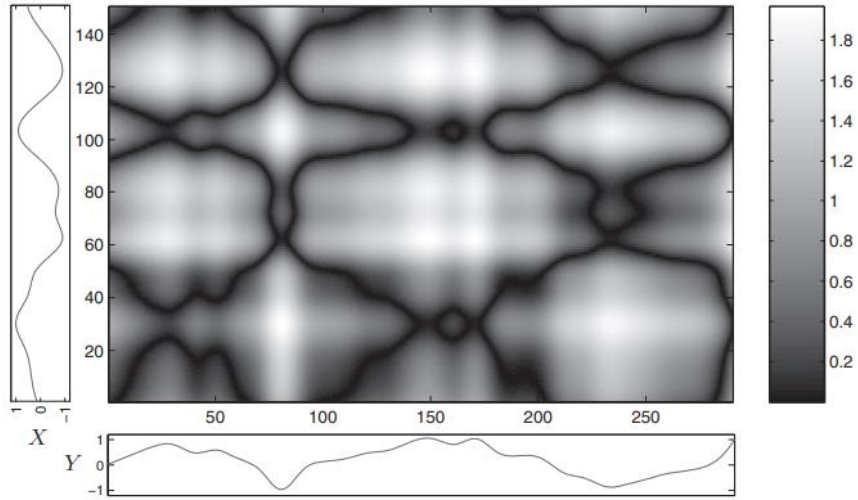
Динамическая трансформация временной шкалы (Dynamic Time Warping)

- Есть две последовательности
  - $X := (x_1,\ x_2, \dots, x_N)$ длиной $N \in \mathbb{N}$
  - Y $:= (y_1,\ y_2, \dots, y_N)$ длиной $Y \in \mathbb{N}$
- Пространство признаков $\mathcal{F}$ где $\mathrm{x}_n, y_m \in \mathcal{F}$, для $\mathrm{n} \in [1: N]$ и $m \in [1: M]$
- Для сравнения нужна оценка расстояния (стоимость) между элементами последовательностей – c
  - $\mathrm{c} : \mathcal{F}x\mathcal{F} \rightarrow \mathbb{R}_{\geq 0}$
  - величина расстояния отражает стоимость между элементами

Динамическая трансформация временной шкалы (Dynamic Time Warping)

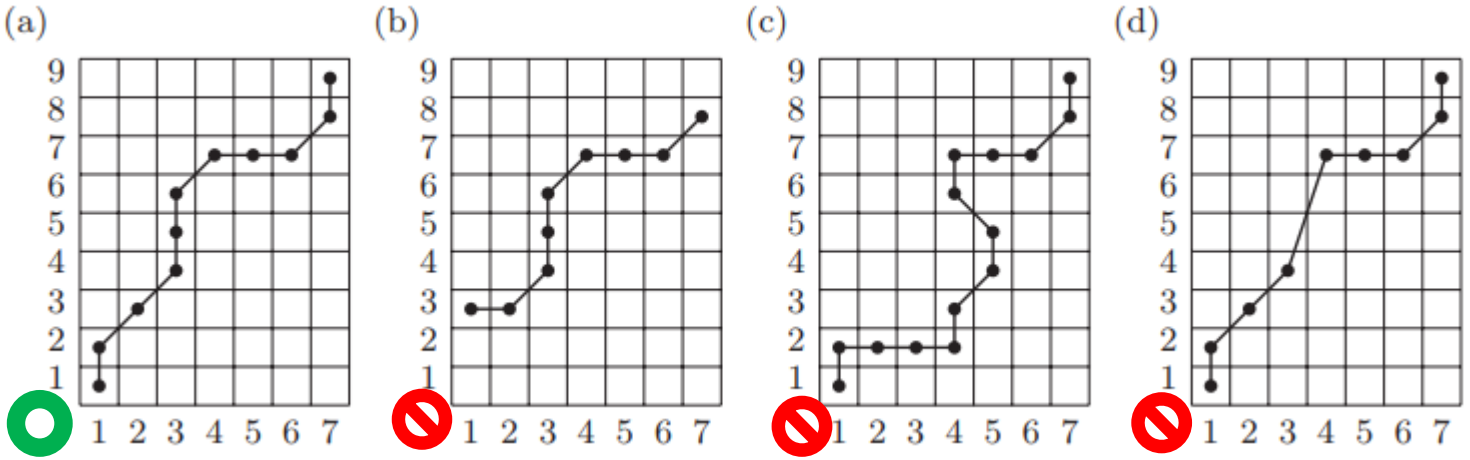
- Оцениваем локальную стоимость для каждой пары элементов X и Y, получим матрицу стоимости
  - $C \in \mathbb{R}^{NxM}$
  - Определена элементами $C(n, m) \coloneqq c(x_n, y_m)$
- Цель – это найти такое выравнивание между X и Y, такое, что суммарная стоимость будет минимальна

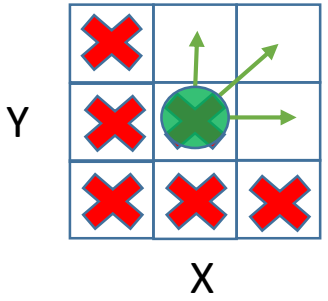




Динамическая трансформация временной шкалы (Dynamic Time Warping)

- Определение:
  - На (N, M) – путь трансформации это последовательность $p = (p_1, \ \dots \ , \ p_L)$ с элементами $p_l = (n_l, m_l) \ \in [1{:}\,N]x[1{:}\,M]$ для $l \ \in [1{:}\,L]$
- Условие:
  - Граничное условие: $p_1 = (1,1)$ и $p_L = (N, M)$
  - Условие монотонности: $n_1 \leq n_2 \leq \ \dots \leq n_L$ и $m_1 \leq m_2 \leq \ \dots \leq m_L$
  - Условие размера шага $p_{l+1} - p_l \ \in \{(1,0), (0,1), (1,1)\}$



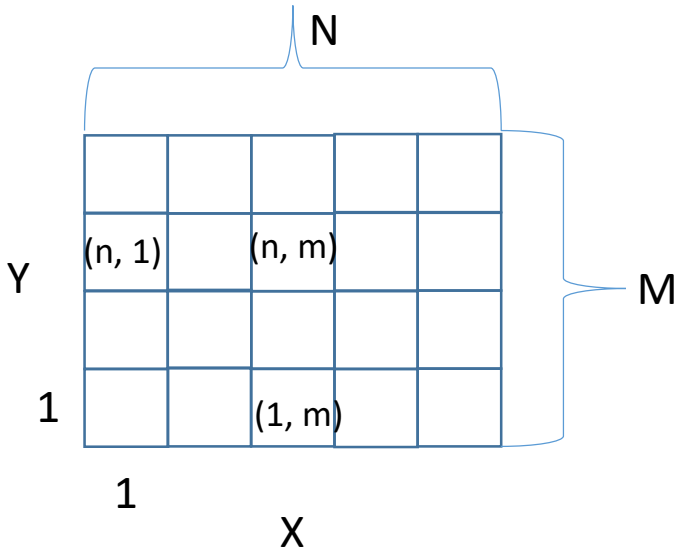


Динамическая трансформация временной шкалы (Dynamic Time Warping)

- На (N, M) путь трансформации $p = (p_1, \ \ldots, p_L)$ определяется выравниванием двух последовательностей X и Y, присваивая элементу x_{n_l}из X элемента y_{m_l} из Y
- Стоимость пути трансформации:
  - $c_p(X, Y) \coloneqq \sum_{l=1}^{L} c\left(x_{n_l}, y_{m_l}\right)$
- Оптимальный путь p^* - это путь с минимальной стоимостью среди всех возможных путей.
- Расстояние трансформации (DTW distance) DTW(X, Y) – общая стоимость пути p^*
  - $DTW(X, Y) = c_{p*}(X, Y) =$ $\min\{c_p(X, Y) | \ p$ есть иуть трансформации на $N, M\}$

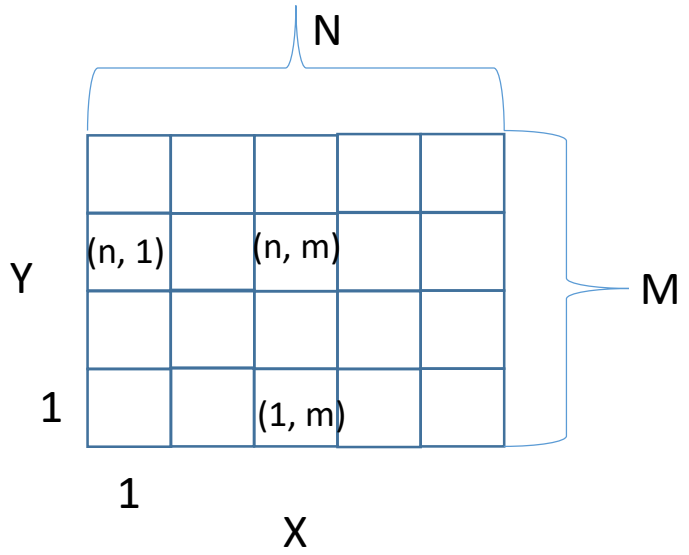
Динамическая трансформация временной шкалы (Dynamic Time Warping)

- Полный перебор всех путей – экспоненциальный рост от длинны M и N
- Используя динамическое программирование можно получить алгоритм со сложностью O(NM)
- Определим префиксные последовательности
  - $X(1:n) \coloneqq (x_1, \dots x_n), n \in [1:N]$
  - $Y(1:m) \coloneqq (y_1, \dots y_m), m \in [1:M]$



Динамическая трансформация временной шкалы (Dynamic Time Warping)

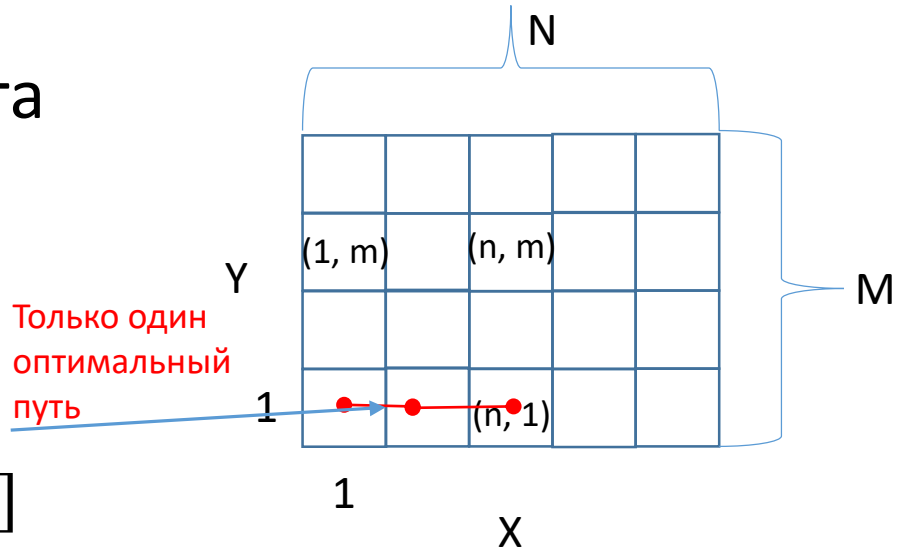
- Полный перебор всех путей – экспоненциальный рост от длинны M и N
- Используя динамическое программирование можно получить алгоритм со сложностью O(NM)
- Определим префиксные последовательности
  - $X(1:n) \coloneqq (x_1, \dots x_n), n \in [1:N]$
  - $Y(1:m) \coloneqq (y_1, \dots y_m), m \in [1:M]$
- И набор значений:
  - $D(n,m) \coloneqq DTW(X(1:n), Y(1:m))$



Значения $D(n,m)$ образуют матрицу накопленной стоимости.
Очевидно, что D(N,M) = DTW(X, Y).

Динамическая трансформация временной шкалы (Dynamic Time Warping)

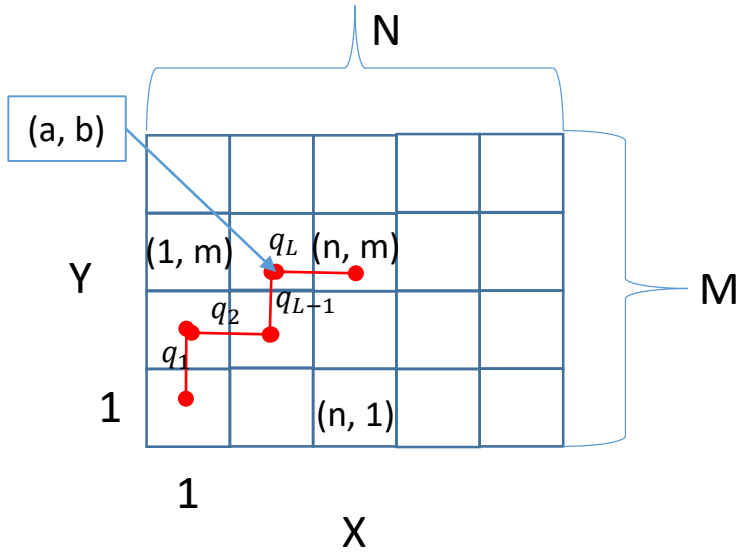
- Теорема об эффективности подсчета значений $D(n, m)$
  - Матрица накопленной стоимости удовлетворяет тождествам:
  - $D(n, 1) \coloneqq \sum_{k=1}^{n} c(x_k, y_1)\ , n\ \in [1: N]$
  - $D(1, m) \coloneqq \sum_{k=1}^{m} c(x_1, y_m)\ , m\ \in [1: M]$
  - $D(n, m) \coloneqq \min\{D(n-1, m-1), D(n-$



Допустим что m = 1 тогда существует только один оптимальный путь между Y(1,1) и X(1, n) со стоимостью $\sum_{k=1}^{n} c(x_k, y_1)$

Динамическая трансформация временной шкалы (Dynamic Time Warping)

- Теорема об эффективности подсчета значений $D(n,m)$
  - Матрица накопленной стоимости удовлетворяет тождествам:
  - $D(n,1) \coloneqq \sum_{k=1}^{n} c(x_k, y_1)\ , n\ \in [1:N]$
  - $D(1,m) \coloneqq \sum_{k=1}^{m} c(x_1, y_m)\ , m\ \in [1:M]$
  - $D(n,m) \coloneqq \min\{D(n-1, m-$



Если n > 1 и m > 1 и существует оптимальный путь $q = (q_1, \dots q_{L-1}, q_L)$ для X(1:n) и Y(1:m). Установим $(a,b) \coloneqq q_{L-1}$, из условия размера шага $(a,b) \in$ {(n − 1, m − 1),(n − 1, m),(n, m − 1)}, и q_{L-1} - оптимальный. Поскольку $D(n,m) = c_{(q_1,..,q_{L-1})}\big(X(1:a), Y(1:b)\big) + c(x_n, y_m)$ то из оптимальности q следует третье тождество

Динамическая трансформация временной шкалы (Dynamic Time Warping)

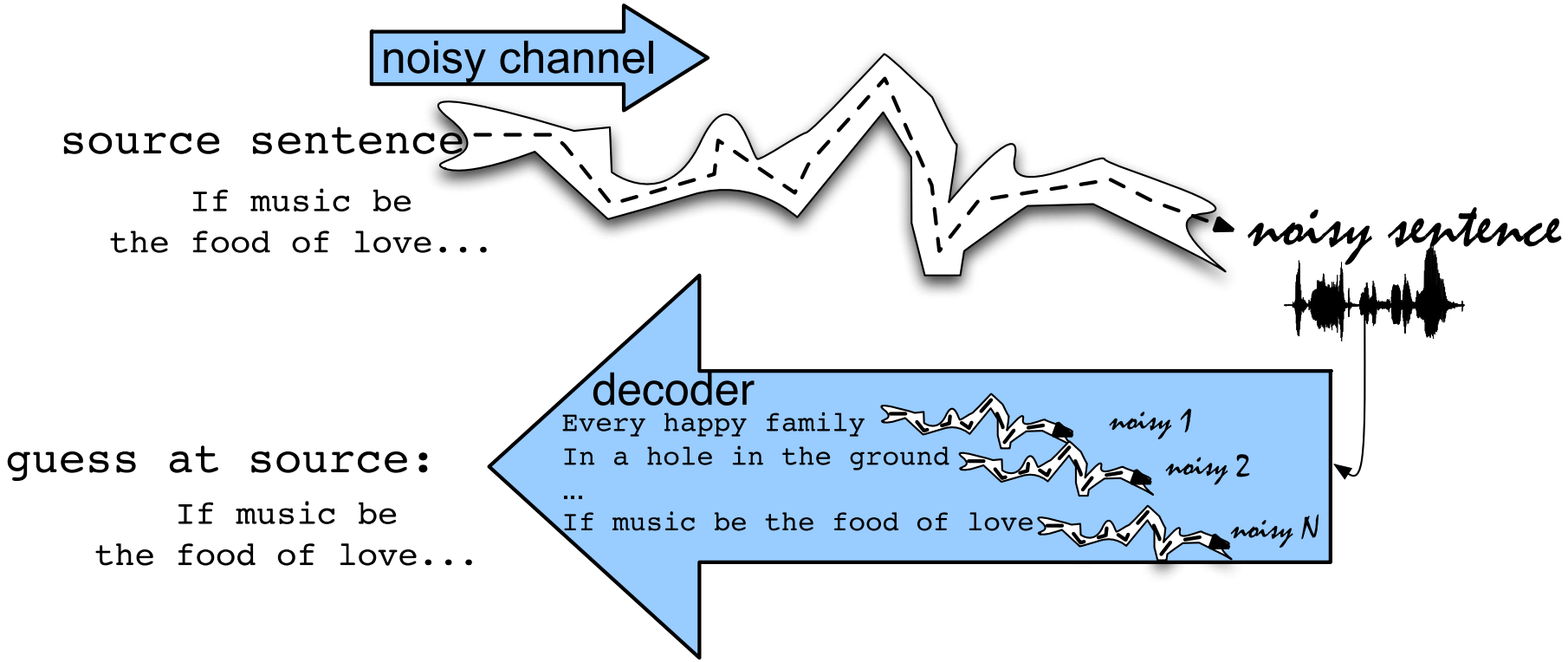
- Алгоритм
  - Вход матрица накопленной стоимости D
  - Выход оптимальный путь p^*
  - Оптимальный путь $p^* = (\mathrm{p}_1, \ldots, \mathrm{p}_\mathrm{L})$ подсчитывается реверсом начиная от $\mathrm{p}_\mathrm{L} = (N, M)$. Считаем элемент $\mathrm{p}_\mathrm{l} = (n, m)$ до (n, m) = (1, 1)

$$p_{\ell-1} := \begin{cases} (1, m-1), & \text{if } n = 1 \\ (n-1, 1), & \text{if } m = 1 \\ \operatorname{argmin}\{D(n-1, m-1), & \\ \qquad D(n-1, m), D(n, m-1)\}, & \text{otherwise,} \end{cases}$$

```
int DTWDistance(s: array [1..n], t: array [1..m]) {
    DTW := array [0..n, 0..m]

    for i := 1 to n
        DTW[i, 0] := infinity
    for i := 1 to m
        DTW[0, i] := infinity
    DTW[0, 0] := 0

    for i := 1 to n
        for j := 1 to m
            cost := d(s[i], t[j])
            DTW[i, j] := cost + minimum(DTW[i-1, j  ],    // insertion
                                        DTW[i  , j-1],    // deletion
                                        DTW[i-1, j-1])    // match

    return DTW[n, m]
}
```

Модель зашумленного канала

- Декодер в виде поиска по пространству всех возможных предложений.



Модель зашумленного канала

- Каково наиболее вероятное предложение из всех предложений на языке $\mathcal{L}$ при некотором акустическом вход *O*?
- Рассмотрим акустический вход *O* как последовательность индивидуальных наблюдений
  - $O = o_1, o_2, \ldots, o_t$
- Аналогично определим предложение как последовательность слов
  - $W = w_1, w_2, \ldots, w_n$

Модель зашумленного канала

- Вероятностная реализация - выбрать последовательность с максимальной вероятностью

$$\widehat{W} = \underset{W \epsilon \mathcal{L}}{\operatorname{argmax}} P(W|O)$$

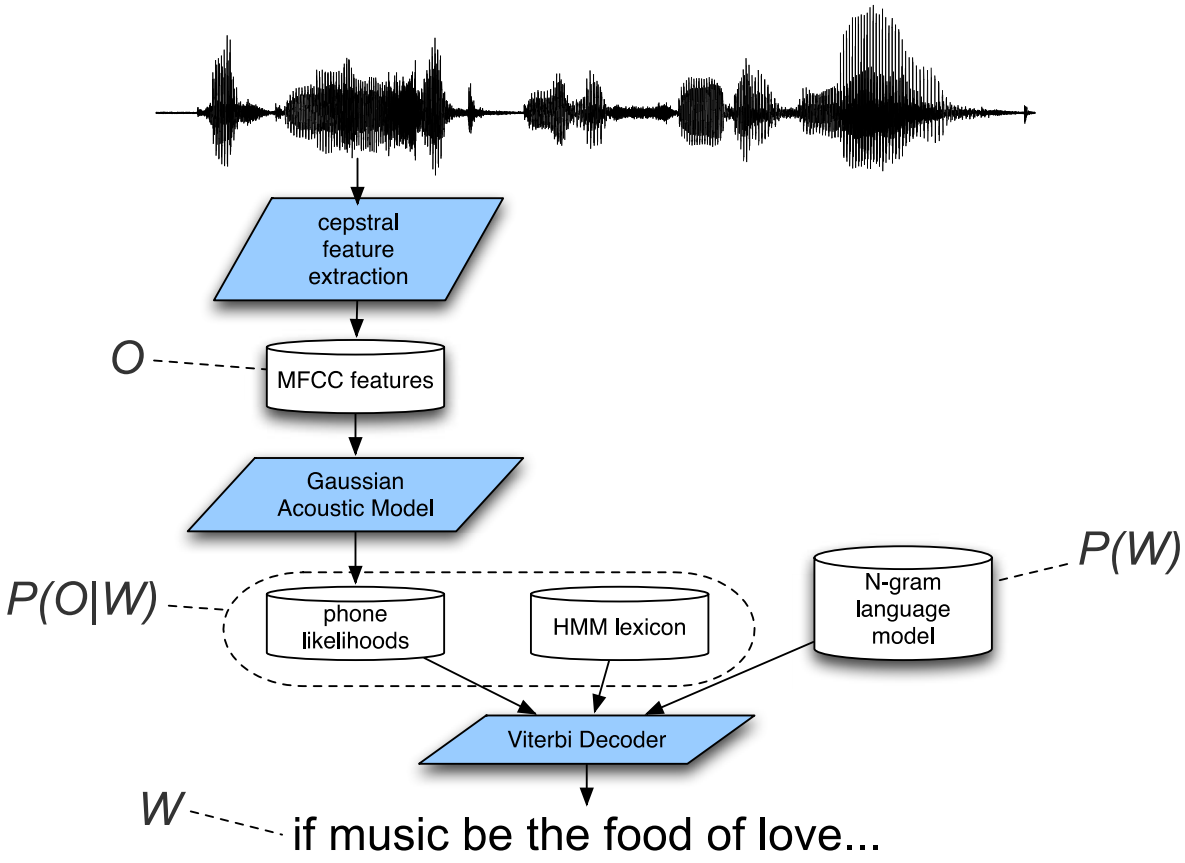
- Используем правило Байеса:

$$\widehat{W} = \underset{W \epsilon \mathcal{L}}{\operatorname{argmax}} \frac{P(O|W)P(W)}{P(O)}$$

- $P(O)$ – одинаковый для всех предложений языка => можем проигнорировать

$$\widehat{W} = \underset{W \epsilon \mathcal{L}}{\operatorname{argmax}} P(O|W)P(W)$$

Архитектура распознавания речи



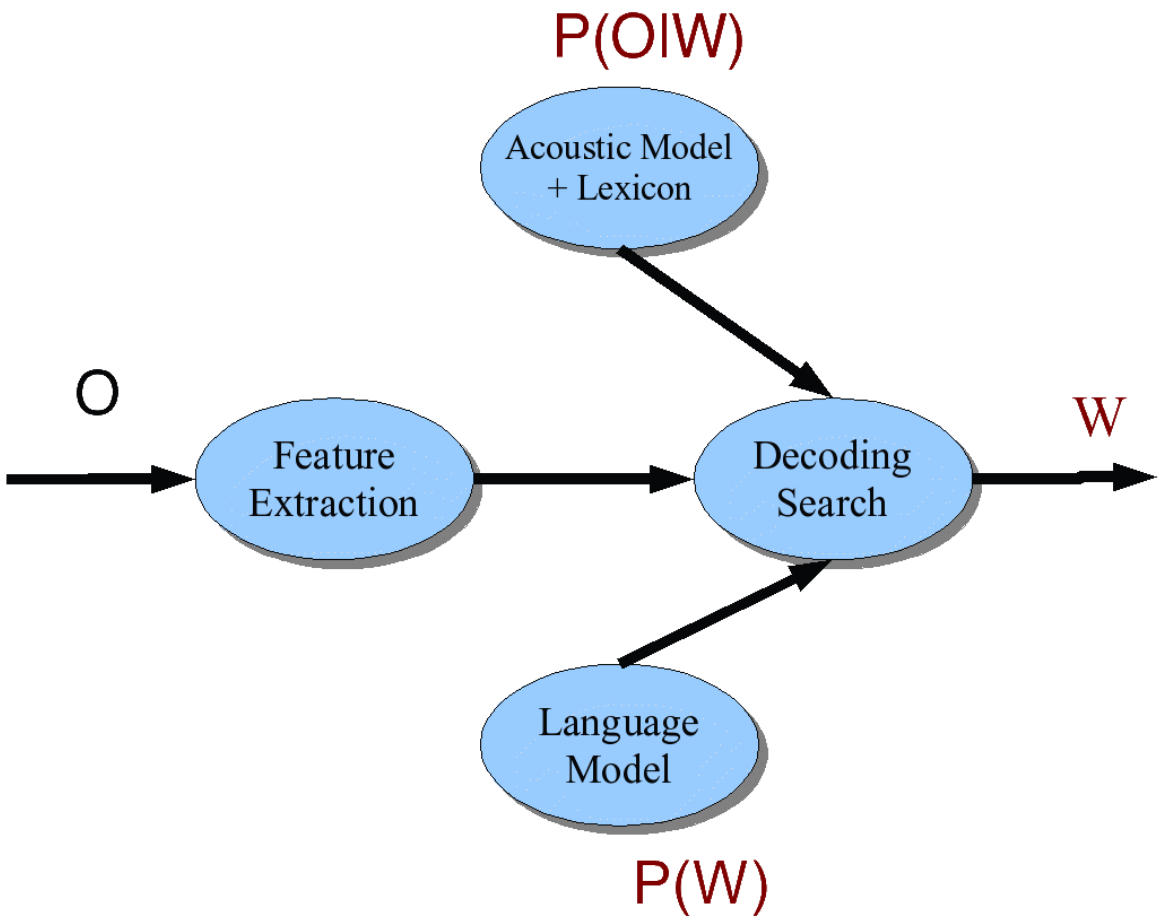
likelihood prior

$$\hat{W} = \underset{W \in \mathcal{L}}{\operatorname{argmax}} P(O|W) P(W)$$

Акустическая модель. Выражает правдоподобие предложения и наблюдаемых значений

Языковая модель. Говорит вероятность встретить предложение в языке

Модель зашумленного канала + архитектура распознования



Декодирование

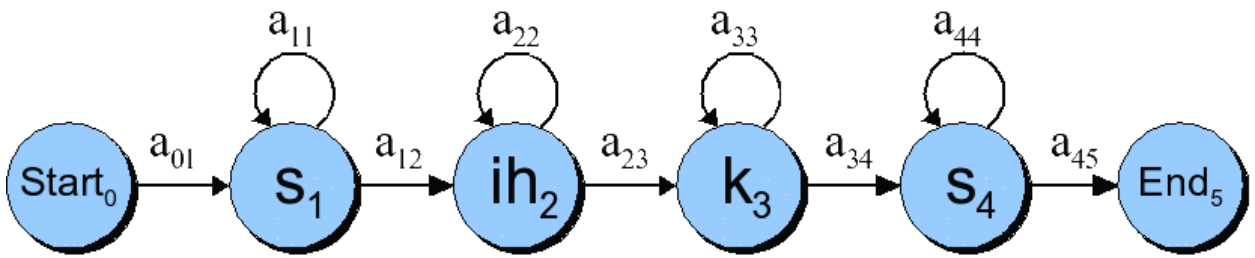
- Извлечение признаков:
  - 39 “MFCC” признаков
- Акустическая модель:
  - Гессианы для вычисления $p(o|q)$
- Модель - Словарь/Произношение
  - HMM: какие фонемы могут следовать друг за другом
- Языковая модель
  - N-grams для расчета $p(w_i|w_{i-1})$
- Декодер
  - Алгоритм Витерби: динамическое программирование для объединения всех этих функций для получения последовательности слов из речи

Словарь

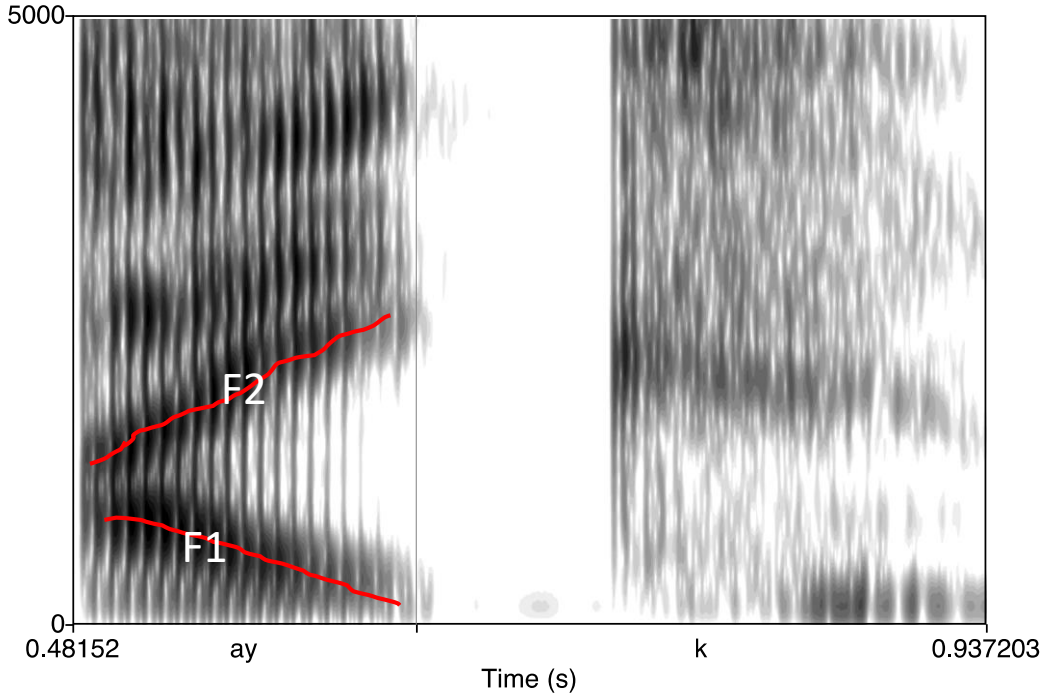
- Список слов
- Каждое слово – это произношение в терминах фонем (фонема - кратчайшая звуковая единица)
- Словарь CMU: 127K слов
  - http://www.speech.cs.cmu.edu/cgi-bin/cmudict
- Мы будем представлять словарь как последовательность состояний HMM

HMM для задачи распознавания цифр

- Наблюдения – это последовательность акустических признаков
- Для задач распознавания используют понятие фонемы
  - Каждая фонема представляет состояние скрытой Марковской модели



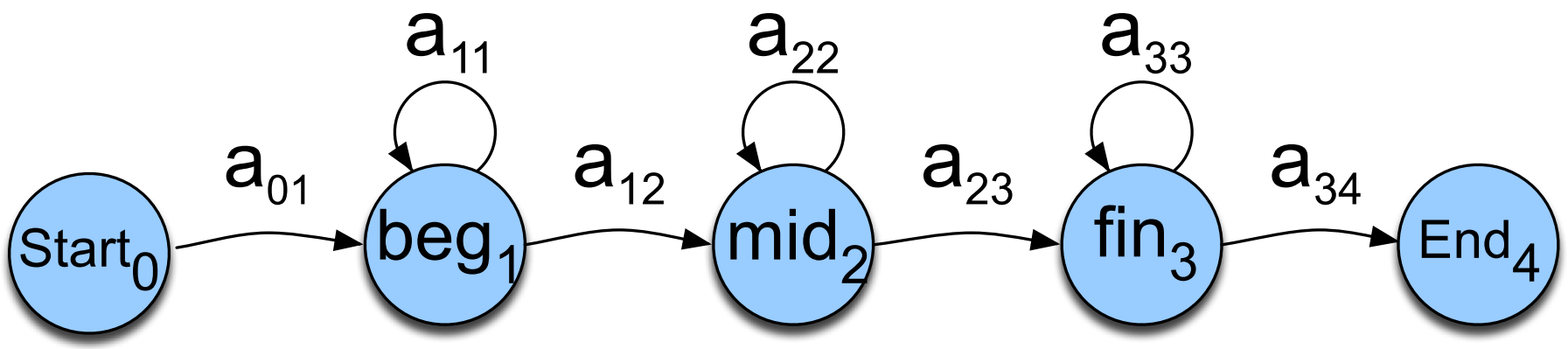
Фонемы не однородны



- Две фонемы в слове “ike”
  - [ay k]
- Форманты гласной фонемы ay F1 и F2 идут в разном направлении
  - F1 – падает
  - F2 – снижается

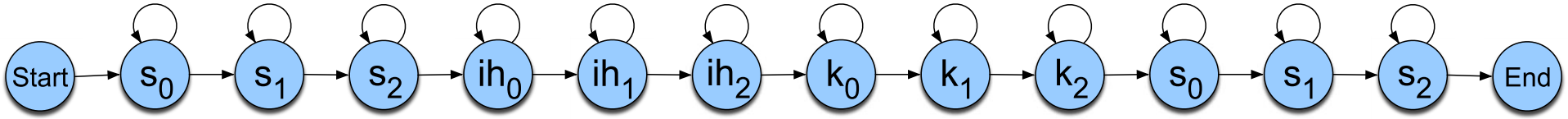
Каждую фонему делим на три части

- Каждая фонема может иметь три под-фонемы:
  - начало
  - середина
  - конец

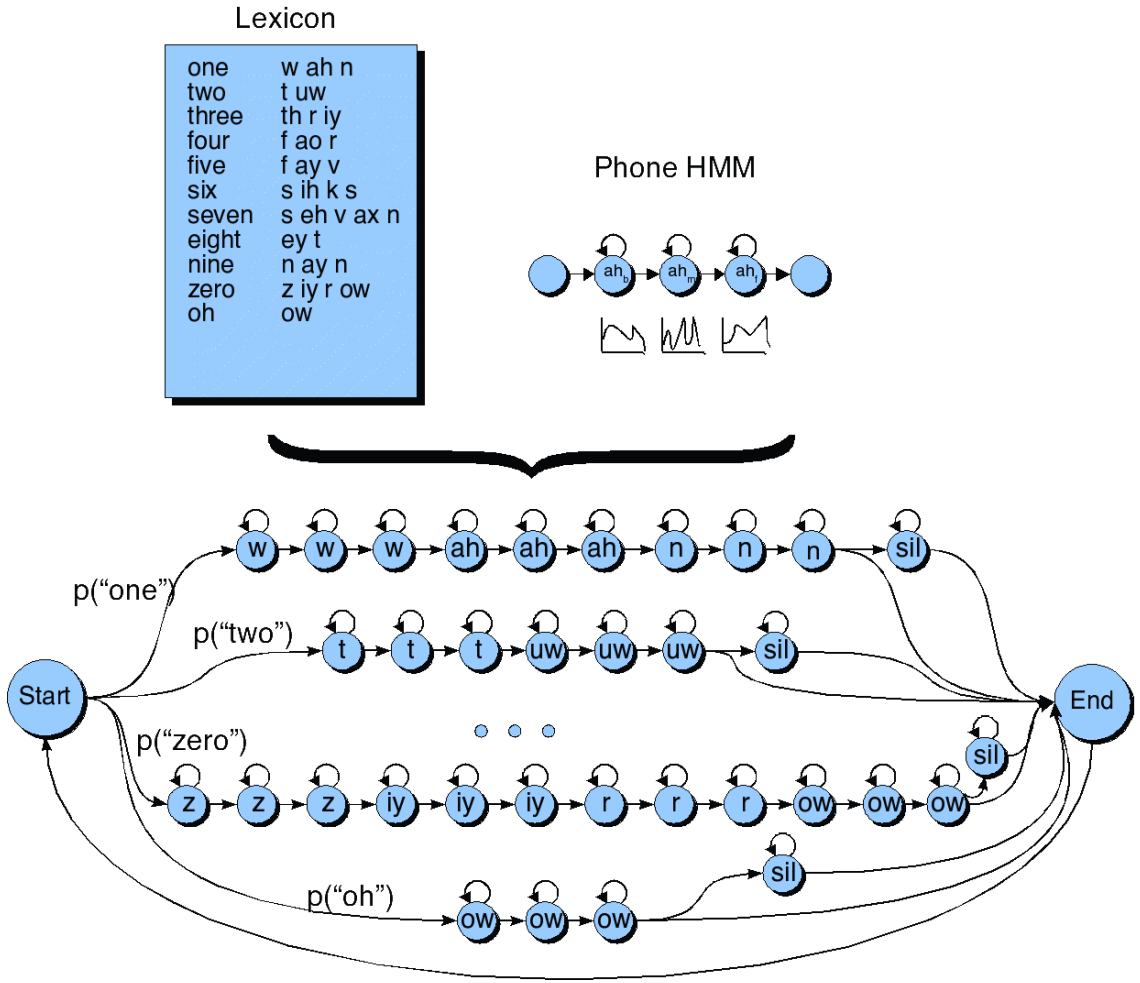


HMM для задачи распознавания цифр

- Результирующая HMM модель слова будет выглядеть



HMM для распознавания цифр



- A и Q представляют словарь (lexicon) - набор:
  - Произношений для слов
  - Каждая фонема имеет набор под-фонем - Q
  - Порядок фонем определяет вероятности перехода A

Компоненты HMM

- $\mathrm{Q} = q_1, \ldots, q_N = N$ – состояний модели
- $\mathrm{V} = v_1, \ldots, v_V$ - алфавит или словарь наблюдений
- $\Pi = \pi_1, \ldots, \pi_N$ – вероятности начальных состояний, π_i - означает, что Марковская модель начинается в состоянии i. $\sum_{i=1} \pi_i = 1$
- $\mathrm{A} = a_{11}, a_{12}, a_{1n}, \ldots, a_{nn}$, - вероятности перехода из состояния i в j $\sum_{j=1}^{n} a_{ij} = 1$
- $\mathrm{B} = b_j(o_t),$ – вероятность пронаблюдать “символ” o_t находясь в состоянии j
- $O = o_1 \ \ldots \ o_T -$ выходная последовательность, наблюдения
- $q_0 \ q_F$ - специальные начальное и финальное состояния. Выход из начального состояния a_{01}, a_{02}, a_{0n}. Вход в финальное состояние a_{1F}, a_{2F}, a_{nF}

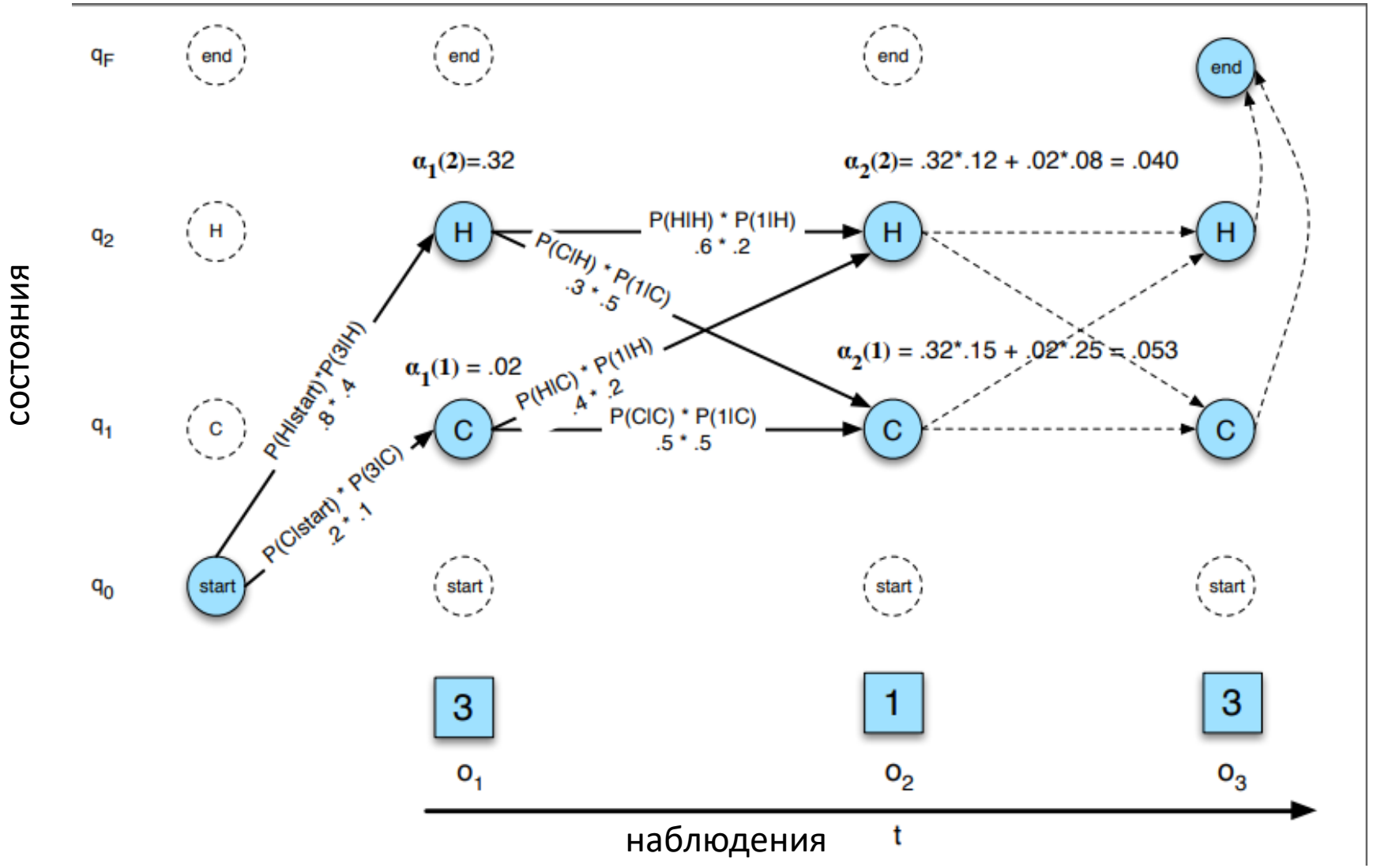
Три задачи HMM

1. Дано: модель $\lambda = (A, B, \Pi)$ и наблюдения O.
   - Оценить насколько наши наблюдения подходят под модель, т.е. оценить вероятность $\mathrm{P}(O|\lambda)$
2. Дано: модель $\lambda = (A, B)$ и наблюдения O.
   - Выбрать последовательность скрытых состояний Q, которая лучше описывает наши наблюдения
3. Дано: последовательность наблюдений O и набор состояний
   - Найти модель, лучше описывающую наши наблюдения $\lambda(A, B, \Pi)$

Задача 1

- *Для оценки нам нужна совместная вероятность двух событий наблюдений и состояний.*
- $P(O,Q) = P(O|Q)P(Q) = \prod_{i=1}^{T} P(o_i|q_i) \prod_{i=1}^{T} P(q_i|q_{i-1})$
- Расчет в лоб - $O(N^T)$ операций
- Динамическое программирование и прямой проход
  - На каждом шаге аккумулируем максимальную вероятность в узлах решетки

Задача 1. Прямой проход



Задача 1

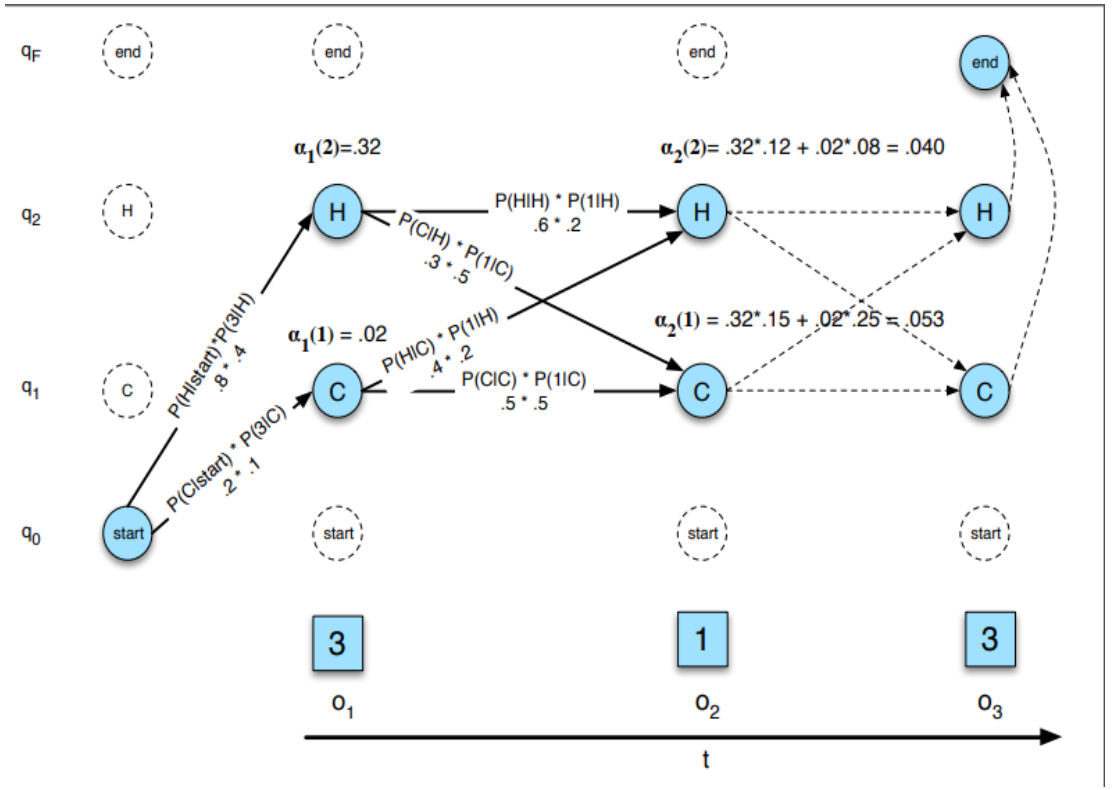
- $\alpha_t(j)$ - вероятность нахождения в узле j – после наблюдения первых t наблюдений задаваемых моделью λ

Каждый элемент сетки выражает следующую вероятность:

$$\alpha_t(j) = P(o_1 o_2 \ \ldots o_t, s_t = j \mid \lambda)$$

или

$$\alpha_t(j) = \sum_{i=1}^{N} a_{t-1}(i) a_{ij} b_j(o_t)$$



Прямой проход алгоритм

1. Инициализация:
   - $\alpha_1(j) = a_{0j} b_j(o_1), \ 1 \leq i \leq N$
2. Рекурсивно считаем:
   - $\alpha_t(j) = \sum_{i=1}^{N} a_{t-1}(i) a_{ij} b_{jo_t}, 1 \leq t \leq T, 1 \leq j \leq N$
3. Окончание
   - $P(O|\lambda) = \alpha_T(q_F) = \sum_{i=1}^{N} \alpha_T(i) a_{iF}$

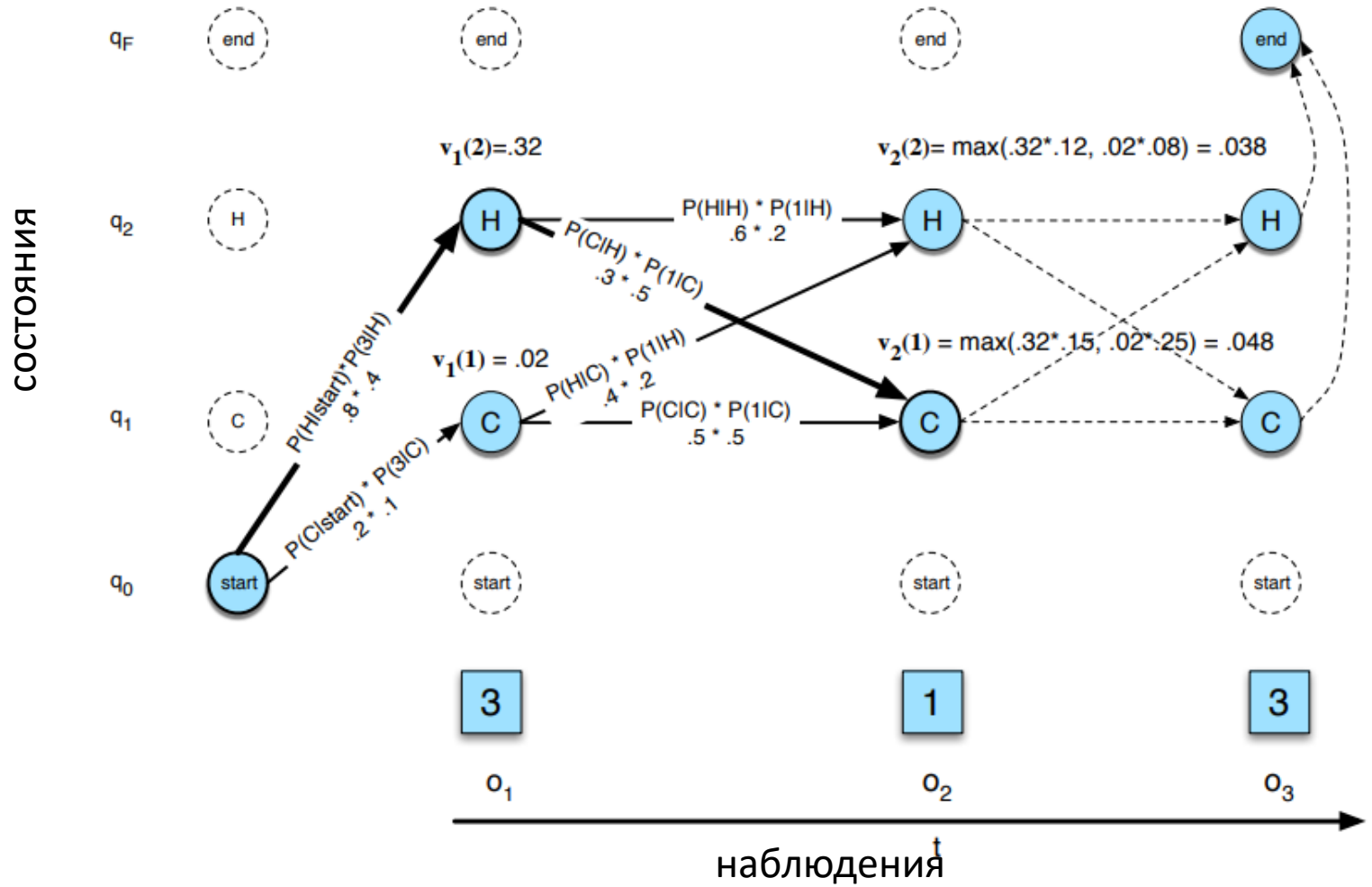
Задача 2 - Декодирование

- Дано
  - модель: $\lambda(A, B)$
  - последовательность наблюдений $O = o_1, o_2, \dots o_T$
- Найти последовательность наиболее вероятных скрытых состояний $\mathrm{Q} = q_1, q_2, \dots, q_T$
- В случае распознавания речи нам нужно по наблюдениям (MFCC фичи) определить последовательность фонем (слов)

Задача 2 – Декодирование

- Наивный подход – запустить прямой проход на перебор состояний,
- Но лучше использовать алгоритм Витерби

Задача 2 – Алгоритм Витерби



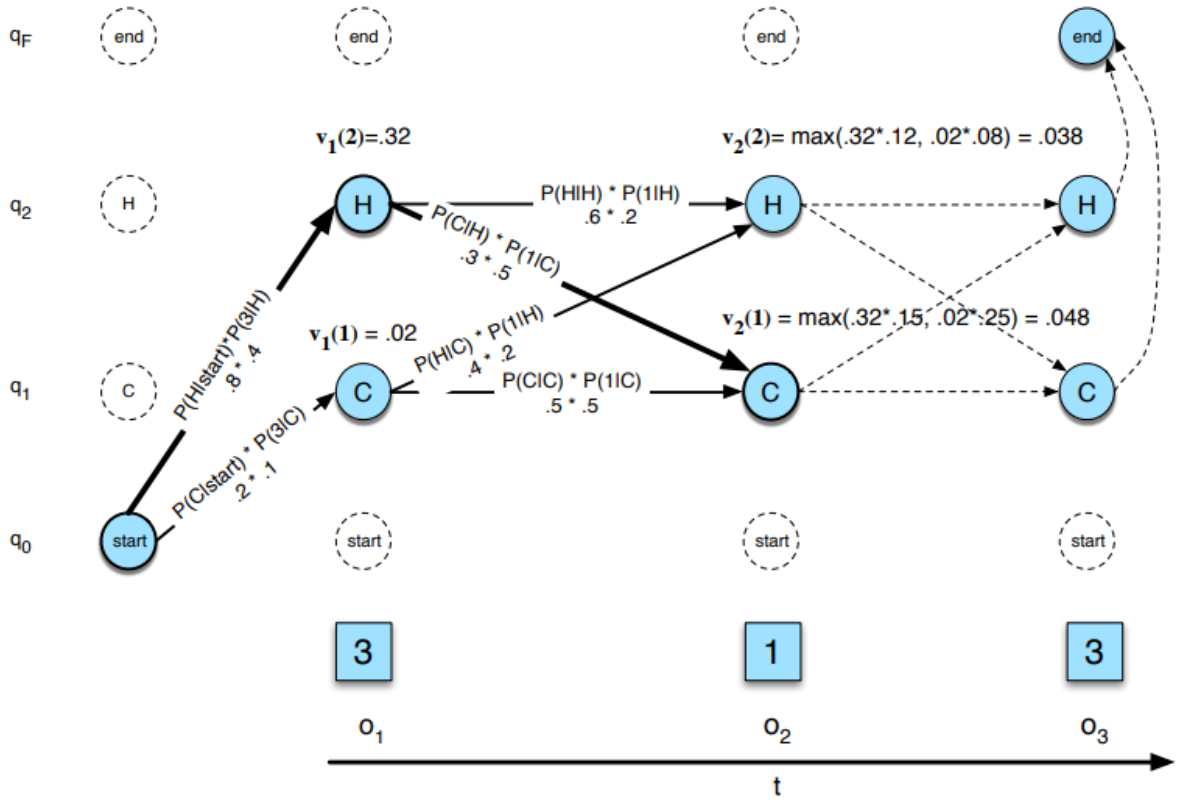
Задача 2 – Алгоритм Витерби

- $v_t(j)$ - показывает вероятность того, что модель находится в наиболее вероятном состоянии *j* после наблюдения *t*

$$v_t(j) = \max_{q_1, q_2, \ldots q_{t-1}} P(q_1, q_2, \ldots q_{t-1}, o_1, o_2, \ldots o_t, q_t = j \mid \mu)$$

или выразив $v_t(j)$ через предыдущее состояние:

$$v_t(j) = \max_{i=1-N} v_{t-1}(j)\,(i) a_{ij} b_j(o_t)$$



Алгоритм Витерби

- Инициализируем:
  - $v_1(j) = a_{0j} b_j, (1), \ \ 1 \leq j \leq N; \psi_1(\mathrm{j}) = []$

Обратный указатель

- Рекурсивно считаем:
  - $v_t(j) = \max\limits_{1 \leq i \leq N} v_{t-1}(i) a_{ij} b_j(o_t), 1 \leq j \leq N, 1 < t \leq T$
  - Сохраняем состояние:
    - $\psi_t(j) = argmax_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t), 1 \leq j \leq N, 1 < t \leq T$
- Заканчиваем, и считываем, что получилось:
  - $\hat{P}_T = v_T(q_F) = \max_{i=1}^{N} v_T(i) a_{iF}$ -финал
  - $q_T = \psi_T(q_F) = argmax_{i=1}^{N} v_T(i) a_{iF}$ - тут путь, который нам нужен
  - $P(\hat{X}) = \max v_T(i)$
- Возвращаем список лучших состояний.

Задача 3 - Обучение

- Дано
  - Последовательность наблюдений $O = o_1, o_2, \dots o_T$
  - Словарь состояний *Q*
- Ищем оптимальные значения модельных параметров: $\mu(A, B)$
- Используем итеративный алгоритм “Вперед Назад” или “Baum-Welch” или EM

Обратный проход

- Инициализация:
  - $\beta_t(i) = a_{1F}, 1 \leq i \leq N$

- Рекурсивно:
  - $\beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(j) a_{ij} b_j(o_t), 1 \leq t \leq T, 1 \leq i \leq N$

- $P(O|\lambda) = a_T(q_F) = \beta_1(q_0) = \sum_{j=1}^{N} a_{0j} b_{jo_1} \beta_1(j)$

Алгоритм “Вперед- назад”

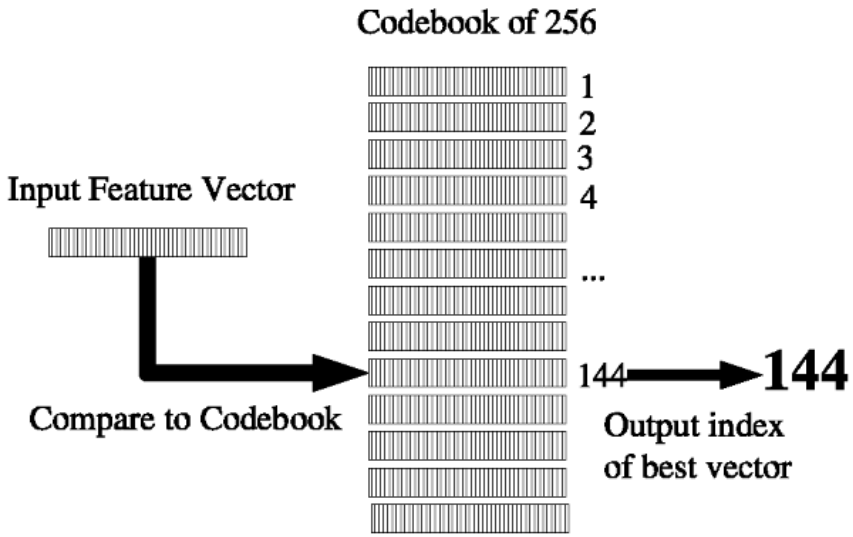
- Начинаем с некоторой модели λ:
  - строим по нашим данным
  - инициализируем равномерно

Шаги:

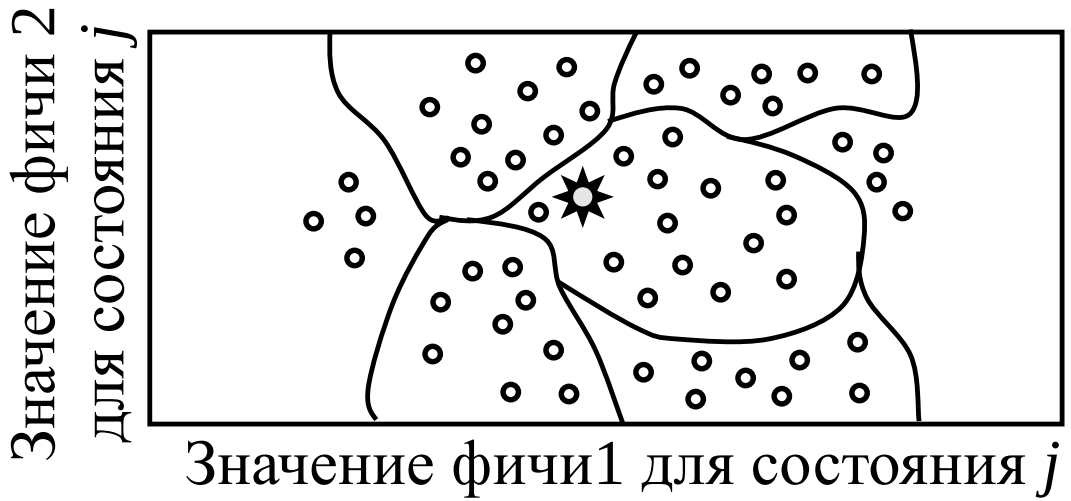
- E: Скармливаем модели наши наблюдения O и оцениваем
  - Вероятность оказаться в состояние j в момент времени t
    - $\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{\alpha_T(q_F)}$; $\forall\ t\ and\ j$
  - Вероятность оказаться в момент времени t в состоянии i, а в момент времени t+1 в состоянии j и пронаблюдать o(t+1)
    - $\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(o_t{+}1)\beta_{t+1}(j)}{\alpha_T(q_F)}$
- M: Затем переоцениваем параметры модели:
  - $\hat{a}_{ij} = \frac{\text{ожидаемое количество переходов из } i \text{ в } j}{\text{ожидаемое количество переходов из } i} = \frac{\sum_{t=1}^{T}\xi_t\,(i,j)}{\sum_{t=1}^{T}\sum_{k=1}^{N}\xi_t(i,k)}$
  - $\hat{b}_{jk} = \frac{\text{ожидаемое количество появлений в } j \text{ с наблюдением } k}{\text{ожидаемое количество появлений в } j} = \frac{\sum_{\{t:o_t=v_k,1\le t\le T\}}\gamma_t(j)}{\sum_{t=1}^{T}\gamma_t(j)}$
- Выход $\lambda(A,B)$

Векторное квантование (Vector Quantization VQ)

- Один из способов соотнести MFCC признаки с символами – это построить функцию мапирования.
  - Кластеризуем вектора
  - Определяем codebook вектор для каждого класса
  - Каждый новый вектор соотносим codebook вектором
  - Выход - индекс вектора
- Это решение не самое лучшее



VQ: Дискретные наблюдения



- $\hat{b}_{jk} = \frac{\text{ожидаемое количество появлений в } j \; codebook \text{ вектора с индексом } k}{\text{ожидаемое количество появлений векторов в } j}$

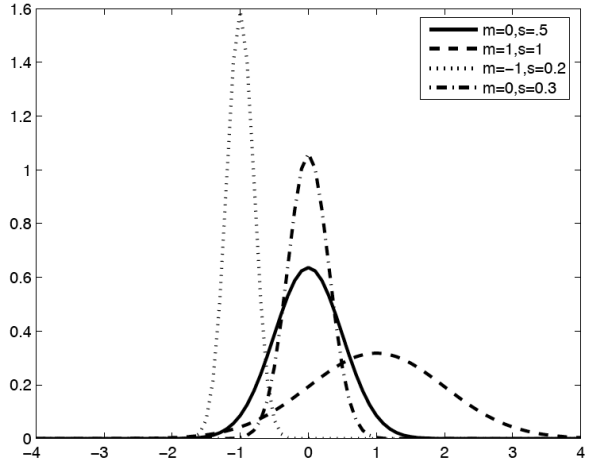
Непрерывные наблюдения

- Одномерный гауссианы
  - Baum-Welch для одномерных гауссиан
- Многомерные гауссианы
  - Baum-Welch для многомерных гауссиан
- Смесь гауссовых моделей (GMMs)
  - Baum-Welch для смеси

Одномерные гауссианы для акустической модели

- Представим, что наш вектор признаков MFCC одномерный
- Тогда мы можем оценить его распределение через μ и σ^2, предполагая, что вектор признаков наблюдений o_t нормально распределен
- Мы можем представить функцию правдоподобия наблюдений $b_j(o_t)$ как гауссиану с параметрами μ и σ^2

$$b_j(o_t) = \frac{1}{\sqrt{2\pi\sigma_j^2}} exp\left(-\frac{(o_t-\mu_j)^2}{2\sigma_j^2}\right)$$



Одномерные гауссианы для акустической модели

- Если мы имеем размеченные данные то:
  - $\mu_i = \frac{1}{T}\sum_{t=1}^{T} o_t\ ;\ \ q_t\ -\ \text{состояние}\ i$
  - $\sigma_i^2 = \frac{1}{T}\sum_{t=1}^{T}(o_t\ -\mu)^2\ ; q_t\ -\ \text{состояние}\ i$
- Но наши данные не размеченные.
  - Но мы можем посчитать вероятность нахождения в состоянии i в момент времени t, через алгоритм Baum-Welch - $\gamma_t(i)$

$$\hat{\mu}_i\ =\ \frac{\sum_{t=1}^{T}\gamma_t(i)o_t}{\sum_{t=1}^{T}\gamma_t(i)} \qquad \hat{\sigma}_i^2\ =\ \frac{\sum_{t=1}^{T}\gamma_t(i)(o_t-\mu_i)^2}{\sum_{t=1}^{T}\gamma_t(i)}$$

Многомерные гауссианы

- Наш вектор MFCC имеет размерность 39 элементов
- Можно смоделировать наши наблюдения многомерной гауссианой с параметрами $\boldsymbol{\mu}$ и $\sum$

$$b_j(o_t) = \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(o_t - \mu_j)^T \Sigma_j^{-1}(o_t - \mu_j)\right)$$

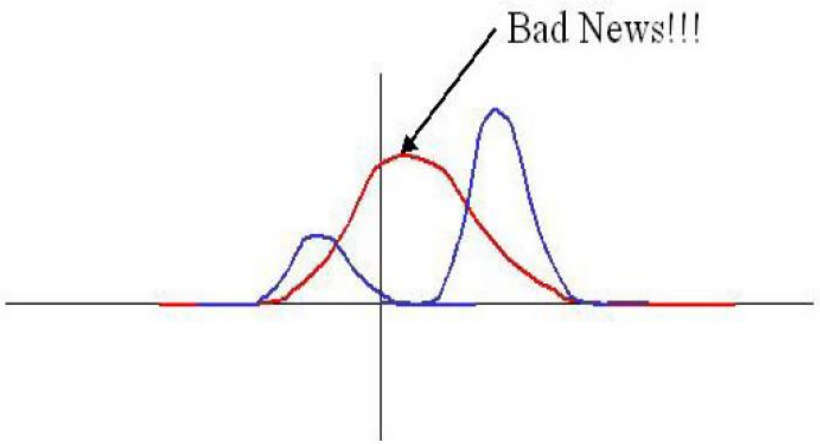
Многомерные гауссианы для акустической модели

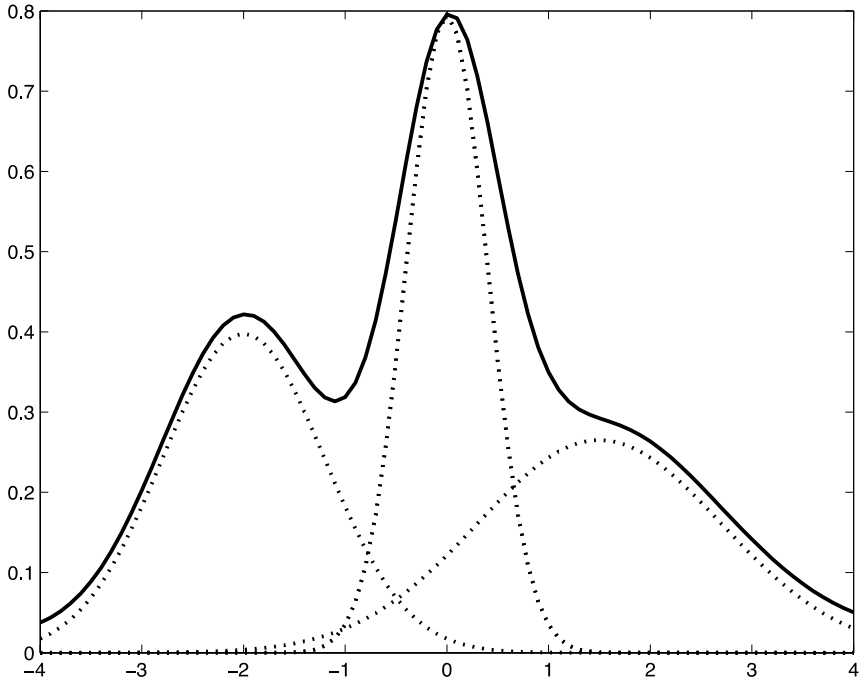
- Оценка параметров модели

$$\hat{\mu}_i = \frac{\sum_{t=1}^{T} \gamma_t(i) o_t}{\sum_{t=1}^{T} \gamma_t(i)} \qquad \hat{\sigma}_i^2 = \frac{\sum_{t=1}^{T} \gamma_t(i)(o_t - \mu_i)(o_t - \mu_i)^T}{\sum_{t=1}^{T} \gamma_t(i)}$$

Смесь гауссиан (GMM)

- Одна гауссиана плохо моделирует распределение





- Выход сделать смесь.

GMM

- Смесь гауссиан – это взвешенная сумма распределений

$$f(x|\mu,\Sigma) = \sum_{k=1}^{M} c_k \frac{1}{\sqrt{2\pi|\Sigma_k|}} exp[(x-\mu_k)^T \Sigma^{-1}(x-\mu_k)]$$

- Тогда определение для функции правдоподобия $b_j(o_t)$

$$b_j(o_t) = \sum_{m=1}^{M} c_{jm} \frac{1}{\sqrt{2\pi|\Sigma_{jm}|}} exp[(x-\mu_{jm})^T \Sigma_{jm}^{-1}(o_t-\mu_{jm})]$$

GMM Оценка параметров

- Оценка параметров модели для каждой компоненты смеси
  - Вероятность нахождения модели в состоянии i в момент времени t

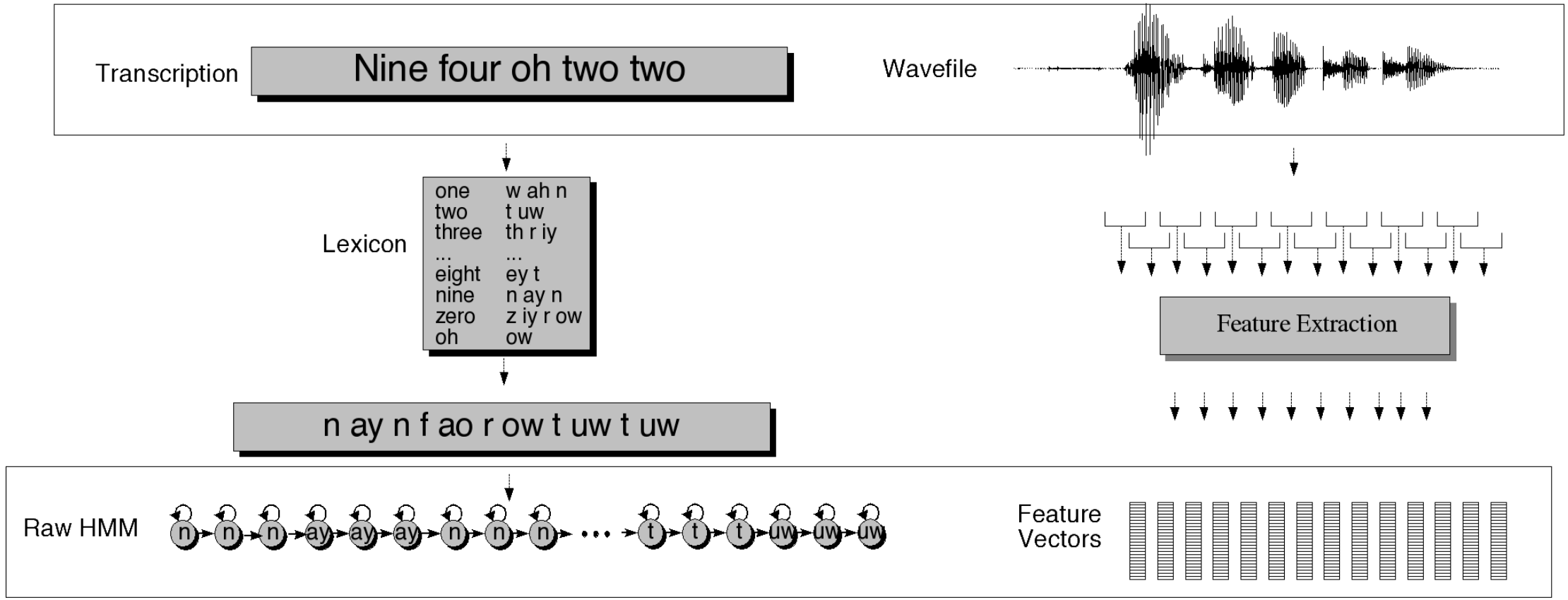
тут нужно считать не вероятность $\gamma_t(j)$, а
вероятность зависимую от компоненты $\gamma_{tm}(j)$

$$\hat{\mu}_{im} = \frac{\sum_{t=1}^{T} \gamma_{tm}(i) o_t}{\sum_{t=1}^{T} \sum_{m=1}^{M} \gamma_{tm}(i)}$$

$$\hat{c}_{im} = \frac{\sum_{t=1}^{T} \gamma_{tm}(i)}{\sum_{t=1}^{T} \sum_{k=1}^{M} \gamma_{tk}(i)}$$

$$\hat{\Sigma}_{im} = \frac{\sum_{t=1}^{T} \gamma_t(i)(o_t - \mu_{im})(o_t - \mu_{im})^T}{\sum_{t=1}^{T} \sum_{k=1}^{M} \gamma_{tm}(i)}$$

Embedded Training



Простая инициализация

- Переходы:
  - Устанавливаем в ноль все нереальные переходы
  - Остальное инициализируем равновероятными значениями
- Правдоподобие:
  - Инициализация μ и σ для каждого состояния глобальными значениями

Embedded Training

- Дано: набор фонем, словарь, транскрибированный звук
  - Строим полный HMM для каждого предложения
  - Инициализируем A матрицу в 0.5, или в 0
  - Инициализируем вероятности B в глобальные средние значения
  - Запускаем итеративно Baum Welch
    - Считаем прямые и обратные вероятности
  - Используем их для переоценки A и B
  - Крутим Baum-Welch до сходимости.

Word Error Rate

- Word Error Rate =

$$\frac{100\ (\text{Insertions}+\text{Substitutions}+\text{Deletions})}{\text{Total Word in Correct Transcript}}$$

```
REF:   portable ****  PHONE UPSTAIRS last night so
HYP:   portable FORM  OF    STORES   last night so
Eval            I     S     S
```

- WER = 100 (1+2+0)/6 = 50%