Цифровая обработка сигналов

Лабораторная работа № 6

Распознавание голосовых команд по образцу

Содержание

1 Теоретические сведения....3
1.1 Мел-шкала....3
1.2 Кепстр и косинусное преобразование....4
1.3 Алгоритм динамической трансформации временной шкалы....5
1.4 Мел-кепстральные коэффициенты....7
1.5 Сопоставление по образцу....8
2 Задание на лабораторную работу....10
2.1 Общая формулировка задания....10
2.2 Инструкции при использовании предоставляемого шаблона lab6.py....11
3 Формат сдачи....12
4 Контрольные вопросы....12

Организация:	Самарский университет
Подразделение:	Кафедра геоинформатики и информационной безопасности
Версия:	2023.12.11-rc

Лифт с голосовым управлением в Шотландии

1 Теоретические сведения

В большинстве случаев сопоставлять голосовые команды во временной области (например, вычислять евклидово расстояние между двумя сигналами) бессмысленно. Это связано с тем, что слух практически не чувствителем к изменениям фазы, то есть два произношения одного и того же слова могут иметь различное представление во временной области. Отчасти это демонстрировалось в лабораторной работе с наложением эффекта роботизации, где обнуление фаз в спектре не особо влияло на возможность распознавания речи.

Таким образом, при сравнении имеет смысл смотреть только на амплитудный спектр. Но делать это нужно с учётом психофизиологических особенностей восприятия звука человеком. Во-первых, воспринимаемая громкость звука пропорциональна логарифму его интенсивности (эмпирический закон Вебера-Вехнера). Во-вторых, субъективно воспринимаемая человеком высота звука сильно зависит от его частоты. Существует эмпирически построенная мел-шкала воспринимаемых высот, которые слушатели воспринимают находящимся на равном расстоянии друг от друга (хотя расстояние по физической шкале частот, измеряемое в герцах, будет неравномерным).

1.1 Мел-шкала

Вариант формулы для перевода f [Гц] в m [мел] (D. O'Shaughnessy, 1987):

$$m = 1127 \ln_{10}\left(1 + \frac{f}{700}\right), \qquad f = 700\left(e^{\frac{m}{1127}} - 1\right).$$

График нелинейной зависимости m от f приведён на рисунке 1.

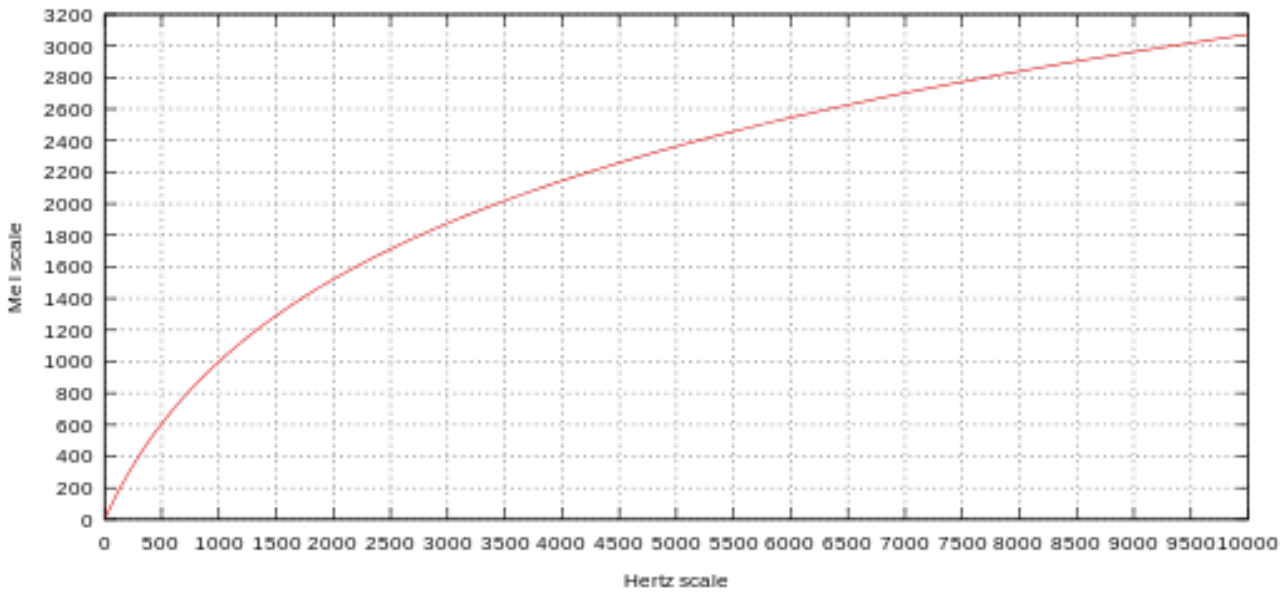

Рисунок 1 — Мел-шкала

1.2 Кепстр и косинусное преобразование

Ранее (на практическом занятии) было доказано равенство Парсеваля: квадрат амплитудного спектра пропорционален энергии сигнала: $\frac{1}{2\pi}\int_{-\pi}^{\pi}|X(e^{j\omega})|^2 d\omega = \sum_n |x[n]|^2$. Аналогичное соотношение верно и для дискретного преобразования Фурье. Таким образом, квадрат модуля спектра может рассматриваться как оценка энергетического спектра (ЭС) — функции, описывающей распределение мощности сигнала в зависимости от частоты. Для дальнейших вычислений будет браться логарифм от ЭС — для учёта того факта, что воспринимаемая громкость звука пропорциональна логарифму его мощности.

В задаче сопоставления по образцу (и во многих других задачах) важно не столько само распределение энергии по частотам, сколько **структура этого распределения**, в частности, наличие в нём периодических структур, соответствующих составным тонам. Напомним, что составной тон — это тон, в котором помимо основной частоты есть ещё кратные ей обертоны, т. е. ЭС такого тона будет иметь повторяющиеся пики. А как выявить периодические структуры в сигнале? Нужно вычислить преобразование Фурье! Прямое или обратное — без разницы, но исторически (?) обычно берут обратное. Таким образом, появляется понятие кепстра — обратного преобразования Фурье от логарифма ЭС (грубо: «спектр от спектра»).

Обратим внимание, что значения ЭС — вещественные. Вычисление преобразования Фурье даст комплексные спектральные значения, что может быть не очень удобным. В этом случае может использоваться дискретное косинусное преобразование (ДКП), которое раскладывает сигнал не на комплексные экспоненты, а на вещественные косинусы. Существует несколько различных форм определения ДКП, вот одна из них:

$$X_{\cos}[m] = \lambda_m \sum_{n=0}^{N-1} x[n] \cos\left(\frac{\pi(2m+1)n}{2N}\right), \qquad \lambda_m = \begin{cases} \sqrt{\frac{1}{N}}, & m = 0; \\ \sqrt{\frac{2}{N}}, & m \neq 0. \end{cases}$$

Отметим пару моментов касаемо вычисления ДКП.

Во-первых, для небольших значений N существуют быстрые алгоритмы вычисления — см., например, статью В. М. Чернова и М. А. Чичёвой «Быстрые алгоритмы дискретных косинусных преобразований коротких длин с минимальной вычислительной сложностью» (Известия Самарского научного центра РАН, 1999, т. 2, с. 241-248).

Во-вторых, для больших значений N ДКП может вычисляться через ДПФ — см., например, статью M. J. Narasimha и A. M. Peterson «On the computation of the discrete cosine transform» (IEEE Transactions on Communications, 1978, т. COM-26(6), с. 934-

936) или пункт 5.2.4 «Совмещённые алгоритмы быстрого дискретного косинусного преобразования» в книге «Методы компьютерной обработки изображений» под ред. В. А. Сойфера (Физматлит, 2003).

1.3 Алгоритм динамической трансформации временной шкалы

Рассмотрим вспомогательный алгоритм, который может использоваться для совмещения двух временных шкал. Отметим, что для наглядности в данном подразделе алгоритм будет рассмотрен на примере обычных целых чисел (одномерный вариант), тогда как в алгоритме распознавания по образцу он будет применяться для вещественных векторов (многомерный вариант; впрочем, различаться будет только способ вычисления веса-расстояния).

Пусть есть две таких последовательности:

$x[n]$:	5	3	1	5	5		9	9	3	1	3	7
$y[n]$:			2	6	4	5		8	2	2		

Пустые ячейки на рисунке нужны только для наглядности представления — какой отсчёт в идеале соответствует какому; реальные сигналы представляют из себя сплошные последовательности чисел:

$x[n] = \{5, 3, 1, 5, 5, 9, 9, 3, 1, 3, 7\}$ (размер последовательности $N = 11$),

$y[n] = \{2, 6, 4, 5, 8, 2, 2\}$ (размер последовательности $M = 7$),

а изображённое на рисунке временное соответствие нам как раз и нужно получить.

При нахождении такого соответствия необходимо учитывать следующие моменты. Во-первых, значения последовательности не совпадают друг с другом точно, лишь приближённо. Во-вторых, последовательности могут быть невыровнены по времени (какие-то части произнесены быстрее, какие-то медленнее). В примере выше желтой подпоследовательности в $x[n]$ в некоторой степени соответствует весь сигнал $y[n]$, при этом двум отсчётам $\{5, 5\}$ соответствует три отсчёта $\{6, 4, 5\}$, а двум отсчётам $\{9, 9\}$ соответствует один отсчёт $\{8\}$. Задача состоит в том, чтобы найти эту самую желтую подпоследовательность — т. е. найти соответствие и получить некоторое число, характеризующее, насколько хорошо (или плохо) найденное соответствие.

Будем полагать, что размер $x[n]$ всегда больше размера $y[n]$, т. е. задача состоит в нахождении в $x[n]$ подпоследовательности, которая наилучшим образом совпадает с $y[n]$.

Для решения этой задачи существует алгоритм динамической трансформации временной шкалы (DTW – Dynamic Time Warping). Выглядит он следующим образом.

Составляется матрица весов $A = \{a_{pq}\}_{p=0,\ q=0}^{M-1,\ N-1}$, где a_{pq} – расстояние (вес, штраф) между элементами $y[p]$ и $x[q]$. Это расстояние может вычисляться различными способами, для нашего одномерного варианта возьмем $a_{pq} = |y[p] - x[q]|$. Тогда для векторов $x[n] = \{5, 3, 1, 5, 5, 9, 9, 3, 1, 3, 7\}$ и $y[n] = \{2, 6, 4, 5, 8, 2, 2\}$ получим:

$$A = \begin{pmatrix} 3 & 1 & 1 & 3 & 3 & 7 & 7 & 1 & 1 & 1 & 5 \\ 1 & 3 & 5 & 1 & 1 & 3 & 3 & 3 & 5 & 3 & 1 \\ 1 & 1 & 3 & 1 & 1 & 5 & 5 & 1 & 3 & 1 & 3 \\ 0 & 2 & 4 & 0 & 0 & 4 & 4 & 2 & 4 & 2 & 2 \\ 3 & 5 & 7 & 3 & 3 & 1 & 1 & 5 & 7 & 5 & 1 \\ 3 & 1 & 1 & 3 & 3 & 7 & 7 & 1 & 1 & 1 & 5 \\ 3 & 1 & 1 & 3 & 3 & 7 & 7 & 1 & 1 & 1 & 5 \end{pmatrix}$$

Путь с минимальным весом по элементам этой матрицы, начинающийся на верхней строке и заканчивающийся на нижней, будет определять наилучшее соответствие, а сумма элементов этого пути может служить характеристикой найденного соответствия (чем оно больше — тем хуже соответствие). При этом данный путь должен удовлетворять следующим ограничениям: он не может идти по матрице влево или вверх, иначе нарушится порядок времени. Так, если $x[q]$ был сопоставлен с $y[p]$, то:

1) $x[q+1]$ может быть сопоставлен с $y[p]$ или $y[p+1]$, но не с $y[p-1]$ (т. е. не может уменьшаться координата p);

2) $y[p+1]$ может быть сопоставлен с $x[q]$ или $x[q+1]$, но не с $x[q-1]$ (т. е. не может уменьшаться координата q).

В простейшем случае такой путь может быть найден рекурсивным путём: перебираем все элементы первой строки и для каждого элемента рекурсивно пытаемся построить путь: смотрим на элементы справа, по диагонали (справа внизу) и внизу.

При более основательном подходе можно использовать классические алгоритмы нахождения кратчайшего пути на графах (надо построить кратчайшие пути от всех элементов первой строки до достижимых элементов последней строки и выбрать из этих путей кратчайший).

В случае, когда у сопоставляемых последовательностей нет никаких сжатий или растяжений друг относительно друга, найденный путь должен быть идеальной диагональю. В нашем примере, так же как и в большинстве реальных случаев, путь будет содержать вертикальные и горизонтальные ступеньки, соответствующие случаям, когда одному отсчёту одной последовательности сопоставляется несколько отсчётов другой последовательности:

$$A = \begin{pmatrix} 3 & 1 & \boxed{1} & 3 & 3 & 7 & 7 & 1 & 1 & 1 & 5 \\ 1 & 3 & 5 & \boxed{1} & 1 & 3 & 3 & 3 & 5 & 3 & 1 \\ 1 & 1 & 3 & 1 & \boxed{1} & 5 & 5 & 1 & 3 & 1 & 3 \\ 0 & 2 & 4 & 0 & \boxed{0} & 4 & 4 & 2 & 4 & 2 & 2 \\ 3 & 5 & 7 & 3 & 3 & \boxed{1} & \boxed{1} & 5 & 7 & 5 & 1 \\ 3 & 1 & 1 & 3 & 3 & 7 & 7 & \boxed{1} & 1 & 1 & 5 \\ 3 & 1 & 1 & 3 & 3 & 7 & 7 & 1 & \boxed{1} & 1 & 5 \end{pmatrix}$$

Итак, позиции элементов пути определяют сопоставление: например, координаты $(p = 0, q = 2)$ первого элемента пути означают, что $y[0]$ сопоставляется с $x[2]$ и т. д. При этом качество найденного соответствия можно охарактеризовать длиной пути (в нашем примере получается 1+1+1+0+1+1+1+1=7) — чем оно выше, тем хуже найденное соответствие.

1.4 Мел-кепстральные коэффициенты

⚠ Все приводимые в данном подразделе значения параметров (выделены фоном) являются ориентировочными, вы можете их варьировать, чтобы попытаться получить более лучший результат.

Для распознавания по образцу используются так называемые мел-кепстральные коэффициенты (MFCC), алгоритм вычисления которых выглядит следующим образом.

1. Для распознаваемого фрагмента сигнала вычисляется STFT. Размер используемого сегмента — 20 мс. Получаем матрицу спектральных комплексных коэффициентов.

2. Каждый столбец в такой матрице представляет собой спектр сегмента. Из каждого такого спектра выделяется набор полос одинаковой ширины по мел-шкале (для учёта субъективности воспринимаемой человеком высоты звука от его частоты). Берётся фиксированное количество (например, 8) полос частот одинаковой ширины по мел-шкале (например, 450 мел). Эти полосы берутся с перекрытием напополам. Далее вычисляется оценка энергии, приходящейся на каждую такую полосу: отсчёты спектра, попадающие в очередную полосу, умножаются на треугольное весовое окно, дальше от каждого значения берётся квадрат модуля и все значения суммируются.

Напомним, что мел-шкала нелинейна относительно герц-шкалы. Например, первой полосе 0...450 мел соответствует диапазон 0...344 Гц. Отсчёты спектра, попадающие в диапазон 0...344 Гц умножаются на треугольное окно, далее от каждого отсчёта берётся квадрат модуля, все полученные значения суммируются — получается одно вещественное число — оценка энергии, приходящейся первую полосу частот.

Операция повторяется для всего набора рассматриваемых полос частот. Обратим внимание, что, например, последней в рассматриваемом примере восьмой полосе 1575...2025 мел соответствует диапазон 2132...3521 Гц. Ширина восьмой полосы составляет 1389 Гц (против 344 Гц у первой полосы). То есть количество отсчётов спектра, попадающих в восьмую полосу будет примерно в 4 раза больше, чем количество отсчётов, попадающих в первую (эффект нелинейности мел-шкалы). Соответственно, каждая полоса частот будет иметь свою собственную ширину на герц-шкале и умножаться на треугольную весовую функцию соответствующего, своего, размера.

Таким образом на этом шаге мы уменьшаем сложность задачи, рассматривая дальше вместо детального ЭС только энергию полос частот. Высота матрицы схлопывается до 8 (количество рассматриваемых полос), и это матрица вещественных значений.

3. Далее учитывается эмпирический закон Вебера-Вехнера — каждое значение матрицы заменяется значением своего логарифма. Обратим внимание, что элементы матрицы могут быть равны нулю или быть очень близкими к нулю, что при вычислении логарифма даст $-\infty$ или большие отрицательные числа. Для компенсации этого эффекта при вычислении логарифма от очередного значения a используют либо конструкцию $\lg(max(a, \epsilon))$, где ϵ — некоторое небольшое число (например, 1e-10), либо конструкцию $\lg(1 + a)$. Каждый столбец полученной матрицы можно трактовать как набор значений, определяющих воспринимаемую громкость соответствующей полосы частот. Данный шаг не изменяет размер матрицы.

4. Далее вычисляется косинусный кепстр — к каждому столбцу применяется ДКП. Разрешается использовать библиотечную функцию `dct`. Данный шаг не изменяет размер матрицы и, таким образом, получается финальная матрица вещественных коэффициентов MFCC. Количество строк в этой матрице определяется количеством рассматриваемых полос частот, количество столбцов — количеством сегментов STFT (т. е. длительностью анализируемого фрагмента).

1.5 Сопоставление по образцу

Для каждого образца из подготовленной базы данных вычисляется матрица коэффициентов MFCC. Обратим внимание, что высота этой матрицы строго фиксирована, а вот ширина может быть произвольной (т. к. определяется длительностью образца).

Далее анализируемый сигнал (будь то файл или поток в реальном времени) разбивается на перекрывающиеся фрагменты. Размер фрагмента нужно взять так, чтобы он по длительности превосходил самый длительный из образцов. Обратите

внимание на следующий компромисс: чем больше размер фрагмента, тем больше вероятность того, что образец попадёт целиком на данный фрагмент; но тем вычислительнее трудоёмче становится алгоритм. В идеале размер фрагмента и перекрытие нужно подобрать таким образом, чтобы самый длительный образец мог целиком попасть на какой-нибудь фрагмент. В качестве отправной точки в качестве размера фрагмента можно взять длительность самого длительного из образцов (с небольшим запасом), а перекрытие делать на 3/4. При таких параметрах гарантированно найдётся фрагмент, на который попадёт как минимум 7/8 самого длительного образца.

Далее для каждого фрагмента вычисляется матрица коэффициентов MFCC. Её ширина должна получиться не меньше ширины матрицы любого из образцов. Затем для каждого образца ищется соответствие между столбцами его MFCC и MFCC фрагмента с помощью алгоритма динамической трансформации временной шкалы (см. подраздел 1.3). Разрешается использовать библиотечную функцию `dtw` с параметром `subseq=True` из пакета `librosa`. В качестве метрики между векторами (столбцами) рекомендуется использовать косинусное расстояние (это расстояние сравнивает направление векторов, игнорируя их модуль; в отличие от эвклидова расстояния, которое используется по умолчанию).

Каждое найденное соответствие даст число (вес пути) — насколько найденное соответствие «плохо». Из всех найденных соответствий необходимо выкинуть (не рассматривать далее) вырожденные решения. Вырожденные решения — это решения, у которых кратчайший путь в матрице весов сильно отличается от диагонального. Такие вырожденные решения соответствуют случаям, когда отрезок (например, длиной 0,5 с) сопоставился с отрезком существенно иной, заведомо неверной, длины (например, 0,1 с или 1,0 с).

Далее из оставшихся невырожденных решений выбирается то, что имеет наименьший вес. Остаётся лишь экспериментальным путём подобрать порог. Если вес найденного соответствия меньше порога, то считается, что образец присутствует на фрагменте. Время начала образца внутри фрагмента можно определить по позиции первого элемента в найденном кратчайшем пути.

Обратите внимание, что из-за наличия перекрытия фрагментов один и тот же образец может обнаружиться неоднократно. Такие дублирующиеся обнаружения можно фильтровать, например, по времени начала образца. В простом варианте выполнения работы такую фильтрацию можно не делать.

2 Задание на лабораторную работу

2.1 Общая формулировка задания

Шаг 1. Записать образцы голосовых команд. Каждая голосовая команда — это одно слово (взять не менее трёх разных слов). Для каждого слова записать несколько образцов (ориентировочно от 6 до 15) его произнесения с разными интонациями и немного различающейся скоростью произнесения. Записывать можно одним файлом с помощью Audacity, этим же ПО можно вырезать из записанного файла отдельные образцы слов.

Шаг 2. Записать анализируемый файл, в котором будут распознаваться слова из предыдущего шага. Произнести не менее 10 слов (в произвольном порядке, можно повторять). Данный файл должен получаться независимой записью, а не путём стыковки образцов из шага 1.

Шаг 3. Реализовать функцию, реализующую алгоритм распознавания по образцу, описанный в пункте 1.5. Данная функция должна принимать базу образцов, подготовленных на шаге 1 и анализируемый файл. Результат работы данной функции — обнаруженные в анализируемом файле слова с метками времени их произнесения. При реализации вы можете (но не обязаны) использовать библиотечные функции `stft`, `dct` и `dtw`, но не можете использовать готовую реализацию `mfcc` (код вычисления мел-кепстральных коэффициентов должен быть написан самостоятельно).

Данное задание может реализовываться разными способами. Рекомендуется использовать первый или третий способ.

Способ № 1 «делаю сам и просто». Вы пишете оригинальную реализацию, не опираясь на предоставленную заготовку `lab6-mfcc.zip`. Наиболее вероятно, она будет работать недостаточно быстро для применения в режиме реального времени (время обработки записи будет сопоставимо или превышать длительность самой записи), однако этого достаточно для выполнения лабораторной работы. Преимущество данного подхода в том, что код будет максимально простым и понятным (хотя и медленным).

Способ № 2 «делаю сам и сложно». Вы пишете оригинальную реализацию, кардинально отличающуюся от предоставленной заготовки `lab6-mfcc.zip`, но работающую достаточно быстро для применения в реальном времени. Но бонусный балл просто за сложность по сравнению с другими способами не предоставляется.

Способ № 3 «делаю по заготовке». Вы используете предоставленную заготовку `lab6-mfcc.zip` и следуете инструкциям из пункта 2.2. Код будет выглядеть слож-

нее, чем при использовании способа № 1, но он будет достаточно быстр для применения в реальном времени.

Первый дополнительный бонусный балл можно получить, если используемая реализация достаточно быстра (способ № 2 или № 3) и для анализа вы берёте не записанный заранее файл, а живой поток с микрофона (можно использовать класс Microphone из пакета SpeechRecognition), который анализируется в режиме реального времени. При этом должна использоваться фильтрация дубликатов распознаваний, возникающих из-за перекрытия анализируемых фрагментов.

Второй и третий дополнительные баллы (в зависимости от сложности) можно получить, если вы используете распознавание команд в режиме реального времени вместе с какой-то полезной семантикой (например, управление в какой-нибудь простенькой программе или игре). Преподаватель может не начислять дополнительные баллы, если ваша работа похожа (всего лишь похожа! — пусть код и оригинальный) на любую уже защищённую работу.

2.2 Инструкции при использовании предоставляемого шаблона lab6.py

⚠ При использовании кода из данной заготовки вы должны досконально разобраться во всём и защищать такй код так же, как свой. Ответ на любой вопрос по любой строке кода по типу «не знаю, что это, взял готовый код из предоставленного шаблона» будет считаться неудовлетворительным. Предоставленный готовый код вы можете модифицировать под себя любым образом.

1. Скачайте подготовленную заготовку `lab6-mfcc.zip`. В распакованной папке создайте директорию `data` и разместите в ней подготовленные образцы фраз. Образцы называть по схеме `записанное_слово-номер.wav`, например, `hello-1.wav`, `hello-2.wav` и т. д. В корневой директории вместе со скриптами разместите контрольный анализируемый файл `test.wav`.

В предоставляемой заготовке присутствуют следующие модули:

`samples.py` – содержит класс `Samples`, представлящий собой базу загруженных образцов и функцию `load_samples` для создания экземпляра этого класса. Данный файл является завершённым и не требует доработки (но вы должны в нём разобраться и можете его модифицировать по своему усмотрению).

`mfcc.py` – должен содержать код для относительно быстрого вычисления MFCC, который необходимо дописать.

`main.py` – должен содержать код для распознавания фраз по образцу в анализируемом файле, который необходимо дописать.

При реализации рекомендуется закомментировать неготовые секции внутри функции `test` в `main.py` и раскомменчивать их по мере готовности.

2. В файле `mfcc.py` реализуйте:

2.1) TODO-1. Используя базу готовых образцов `samples` (а точнее — образец с максимальной длительностью из этой базы) и объект контекста `context`, оцените максимальное количество сегментов STFT (максимальную ширину матрицы MFCC).

2.2) TODO-2. Реализуйте формулу для конвертации мел в герцы.

2.3) TODO-3. Вычислите `window_idx`-ую строку матрицы MFCC по уже данному взвешенному срезу STFT `stft_values`. При реализации необходимо обойтись без циклов (обратите внимание на параметр `axis` у sum).

2.4) TODO-4. Примените ДКП к каждому столбцу полученной матрицы. При реализации необходимо обойтись без циклов (см. параметр `axis` у dct).

3. В файле `main.py` реализуйте TODO-5 – сопоставление по образцу (см. подраздел 1.5).

3 Формат сдачи

Предоставить базу образцов, анализируемый входной `wav`-файл и скрипт(ы) с выполненной лабораторной работой.

Сначала вы демонстрируете работоспособность вашего алгоритма на предварительно записанном входном файле.

Далее, прямо во время сдачи и в присутствии преподавателя, вы записываете второй анализируемый файл и демонстрируете работоспособность вашего алгоритма на свежезаписанном файле.

4 Контрольные вопросы

Алгоритм распознавания по образцу с использованием MFCC-коэффициентов.