Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(национальный исследовательский университет)
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, доцент, к. т. н.
_____К.А. Домбровский
«____»_____2017 г

Разработка анализатора музыкальных предпочтений

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ- 09.03.01.2017.382 ПЗ ВКР

Руководитель проекта, доцент, к. т. н.
_____И. Л. Надточий
«_____»_____2017 г.

Автор проекта
студент группы КЭ-445
_____А. Д. Стихарный
«_____»_____2017 г.

Нормоконтролер, ст. преп. каф.
«Электронные вычислительные
машины»
_____В. В. Лурье
«_____»_____2017 г.

Челябинск 2017

АННОТАЦИЯ

Стихарный А.Д.. Разработка анализатора музыкальных предпочтений. – Челябинск: ФГБОУ ВПО «ЮУрГУ» (НИУ) ВШЭКН; 2017, 108 с., 37 ил.

Библиографический список – 16 наименований.

В рамках выпускной квалификационной работы решается задача классификации музыкальных треков в соответствии с пристрастиями слушателей. Анализ необходимо произвести без использования метаданных, используя исключительно аудио дорожку.

Задача состоит из ряда условий:

1. Коллекция треков у каждого пользователя сравнительно небольшая (100-500 штук).
2. Необходимо провести оценку численных характеристик композиций, не используя никакой информации кроме формы звуковой волны, содержащейся в файле (наподобие имён файлов, альбомов, id3-теги).
3. Подать анализатору на вход треки, привязав их к конкретному пользователю, представляющие собой отдельный класс.
4. Спрогнозировать к какому классу (пользователю) относится композиция, подаваемая на вход анализатору и находящаяся за пределами обучающей выборки.

Для решения этой задачи строится собственный программный комплекс, способный успешно решать данную задачу. Производится детальный анализ методов и характеристик в области аудиоаналитике и их выборка для организации работы системы. Рассматриваются модели решения задачи автоматизированного поиска похожей музыки.

Пояснительная записка к выпускной квалификационной работе оформлена в текстовом редакторе MS Office Word.

					ЮУрГУ-09.03.01.62.2017.382 ПЗ ВКР			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		А.Д. Стихарный			Разработка анализатора музыкальных предпочтений	Лит.	Лист	Листов
Пров.		И.Л. Надточий				Д		
						ФГБОУ ВПО «ЮУрГУ» (НИУ)		
Н. контр.		В.В. Лурье						
Утв.		К.А.Домбровский						

Ниже приведен перечень вопросов (ключевых слов), освещенных в данной работе:

1) FFT, оконные функции, обработка сигналов
2) Нейронные сети. Построение и обучение
3) Элементы математической статистики
4) Нормировка и уменьшение размерности

Оглавление

ВВЕДЕНИЕ 5

1 **АУДИОАНАЛИТИКА** 9

1.1 Методы детектирования аудиособытий. 9

1.1.1 Метод детектирования аудиособытий на основе стандартного отклонения нормированных значений мощностей блоков. 10

1.1.2 Метод детектирования аудиособытий на основе использования медианного фильтра 11

1.1.3 Метод детектирования аудиособытия на основе динамического порога для значений мощностей блоков 13

1.2 Классификация аудиособытий. 14

1.2.1 Буферизация с перекрытием 14

1.2.2 Стадия предобработки 15

1.2.3 Извлечение признаков 16

1.2.4 Постобработка признаков 22

1.2.5 Выбор классификатора 24

1.3 Существующие модели поиска похожей музыки. 29

1.3.1 Рекомендательная технология Диско 29

1.3.2 Функция поиска похожего трека Яндекс.Музыка 32

1.3.3 Система подбора похожего контента Genius 33

1.4 Исследования в области неврологии. 34

2 **ПЛАНИРОВАНИЕ** 37

2.1 Входные данные. 37

2.2 Обработка аудиофайла. 38

2.2.1 Разбиение на фреймы. Предобработка 39

2.2.2 Преобразование Фурье 41

2.2.3 Извлечение признаков. Уменьшение размерности. 43

2.2.4 Формирование вектора признаков. 47

2.3 Классификатор. 48

2.3.1 Выбор структуры ИНС 49
2.3.2 Нормировка. 49
2.3.3 Обучение. 49
3. **РАЗРАБОТКА** 53
3.1 Термины и сокращения. 53
3.2 Основные сведения. 55
3.3 Требования к программе. 55
3.3.1 Требования к GUI 55
3.3.2 Функциональные требования. 56
3.4 Требования к видам обеспечения. 56
3.4.1 Требования к хранению данных 56
3.4.2 Требования к языкам программирования. 56
3.4.3 Требования к производительности. 57
3.4.4 Требования к аудио файлам 57
3.4.5 Требования к аппаратному обеспечению. 57
3.5 Проектирование. 57
4. **РЕАЛИЗАЦИЯ** 61
4.1 Реализация блока сбора статистики. 61
4.2 Реализация аналитического блока. 69
4.3 Реализация связующего блока. 75
4.4 Реализация системы рекомендаций. 76
5. **ОЦЕНКА РЕЗУЛЬТАТОВ РАБОТЫ СИСТЕМЫ** 79
ЗАКЛЮЧЕНИЕ 80
Библиографический список 81
Приложение. 83

ВВЕДЕНИЕ

На основе чартов музыкальных предпочтений слушателей сервиса «Яндекс.Музыка» замечено, что пользователи все чаще ищут новую музыку, а не ставят на повтор проверенные композиции [2]. На сегодня одна из самых сложных и интересных задач для музыкальных сервисов — научиться подбирать музыку под музыкальные пристрастия пользователя.

Задача поиска акустически похожих треков достаточно непростая, потому что понятие «схожести» музыки довольно условно. Для кого-то важно, чтобы был похож вокал, другой услышал интересный музыкальный инструмент, а третьему важен ритм. В качестве инструмента для решения данной проблемы в этой работе используются искусственные нейронные сети (ИНС).

ИНС — один из методов машинного обучения, который стал особенно популярен в последние годы. Нейросети прекрасны тем, что им достаточно показать, условно, что такое хорошо, а что такое плохо, чтобы получить желаемый результат. Данный метод обучения ИНС получил название - метод обучения с учителем. Нейросети доказали свою эффективность в области распознавания изображений. Например, нейронную сеть можно обучить распознавать на изображениях те или иные объекты — скажем, автомобили или собак. В ходе обучения ей показывают огромное количество картинок, где есть нужные объекты (положительные примеры) и где их нет (отрицательные примеры). В результате ИНС получает способность верно определять нужные объекты на любых изображениях. К примеру, система распознавания лиц от компанни "DeepFace" имеет заявленную точность в 97%. Эта система обладает устойчивостью к разной освещенности, различию в фоне и ракурсе лица на изображении.

Хочется отметить, что результаты в области распознавания звуков несколько скромнее. Самой успешной является система "SoundNet" [3]. На наборе данных с 10 различными категориями звуков SoundNet классифицирует звуки с точностью 92%, а на наборе данных с 50 категориями показывает точность 74%. Для сравнения, на тех же наборах данных люди показывают точность распознавания, в среднем, 96% и 81%. Но особенностью этой системы является то, что она использует наработки в области распознавания объектов и сцен по видео. А именно использовался метод естественной синхронизации между машинным зрением и машинным слухом, научив нейросеть автоматически извлекать звуковую репрезентацию объекта с неразмеченного видеоматериала.

Method	Accuracy on ESC-50	ESC-10
SVM-MFCC	39.6%	67.5%
Convolutional Autoencoder	39.9%	74.3%
Random Forest	44.3%	72.7%
Piczak ConvNet	64.5%	81.0%
SoundNet	**74.2%**	**92.2%**
Human Performance	81.3%	95.7%

Рисунок 1. Оценка эффективности распознавания звука различными системами в зависимости от количества категорий

Музыка - одна из древнейших областей искусства. Известно, что музыка оказывает существенное влияние на психофизическое состояние человека. Через музыку в человеке можно пробудить те или иные эмоции. Музыкальные предпочтения, складывающиеся у человека, довольно точно характеризуют эмоциональное состояние, к которому он стремится. Люди часто объединяются в неформальные объединения по музыкальным предпочтениям, прослушивание приобретает статусный характер. В то же время написание и распространение музыки сегодня крупный и быстро развивающийся бизнес. Особенностью этого бизнеса является то, что его рост обеспечен ростом продаж музыки в цифровом виде. Это достигается за счет 3 основных факторов:

1. Борьба с пиратским контентом;

2. Изменение модели потребления музыкального контента: переход на прослушивание музыки посредством мобильных устройств;

3. Изменение потребительского поведения: все больше пользователей готовы платить за контент, особенно в экосистеме iOS.

Многие участники рынка отмечают, что именно возможность распространять музыкальный контент через мобильные платформы является одним из наиболее важных факторов, который повлиял значительно на развитие рынка цифрового музыкального контента. Более того, модель

«подписки» набирает популярность вместе с развитием предложения музыкальных сервисов главных участников рынка, специализирующихся на данном виде услуг.

Рисунок 2. Мировой рынок цифровой музыки в денежном выражении, млрд долл.

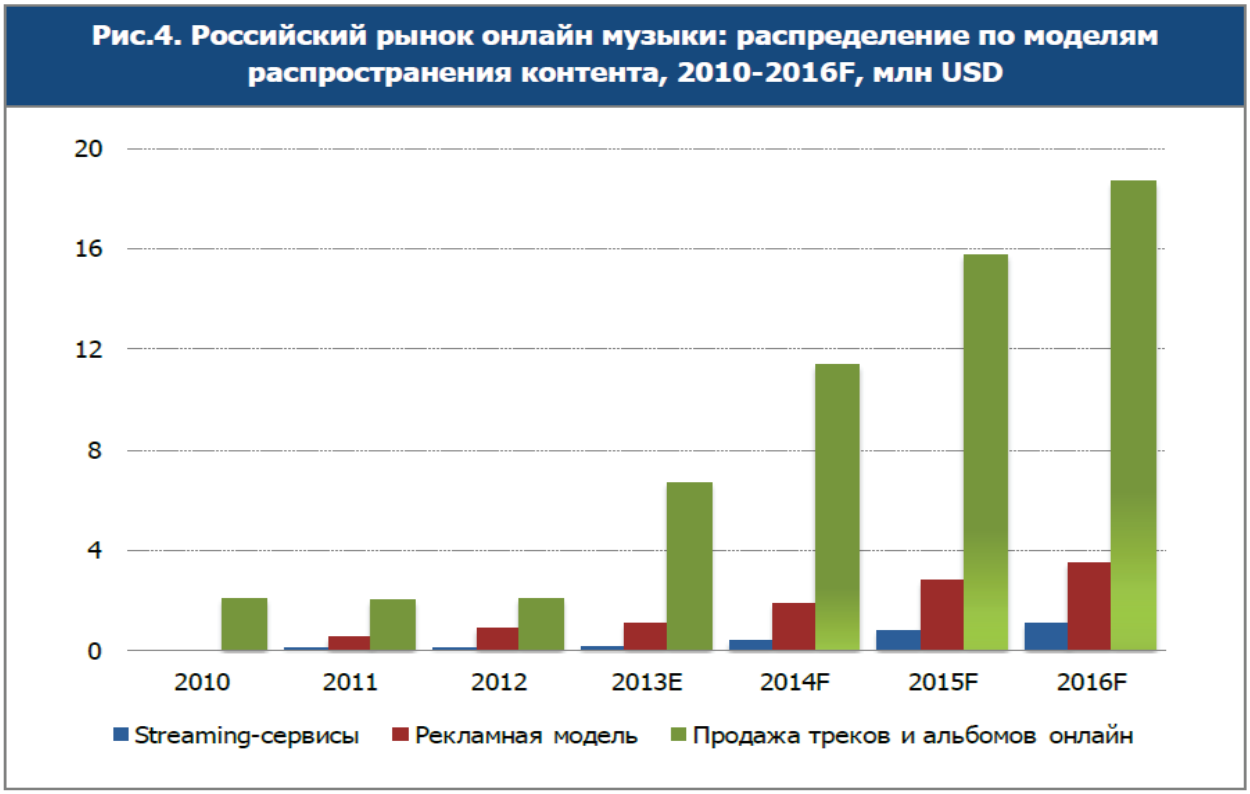


Рисунок 3. Российский рынок онлайн музыки: распределение по моделям распространения контента, 2010-2016 г., млн USD

Актуальность темы, формирования системы подбора аудио контента на основе музыкальных предпочтений заключается в том, что она способна **стимулировать продажи на рынке цифровой дистрибуции,** что весьма интересно для игроков этого рынка.

Автор считает, что исследования в данной области могут принести большую, как в научном, так и в коммерческом плане. **Целью** представленной выпускной квалификационной работы является:

1. Детальный анализ методов, решений и характеристик в области аудиоаналитики;
2. Разработать собственную систему аудиоанализа музыкальных коллекций исключительно на основе звуковой волны, способную у спешно решать поставленную в этой работе задачу;
3. Выборка и анализ результатов работы системы.

Также в этой работе рассматриваются существующие решения, как модели в целом, в области поиска похожих музыкальных композиций.

ГЛАВА 1. АУДИОАНАЛИТИКА

1.1 Методы детектирования аудиособытий

Прежде всего хотелось бы выделить из чего состоит задача распознавания аудиособытий. Она подразделяется на 2 подзадачи:

1. Детектирование (выделение) резких импульсных сигналов из фонового шума в потоке аудио данных;

2. Классификация (распознавание) детектированного сигнала к одному из типов аудио событий.

Большинство методов детектирование импульсных выделяющихся аудиособытий основаны на определении мощности для набора последовательных неперекрывающихся блоков аудио сигнала. Мощность для k-го блока сигнала, состоящего из для N отсчетов, определяется как:

$$e(k)=\frac{1}{N}\sum_{n=0}^{N-1}x^2(n+kN) \qquad k=0,1,....$$

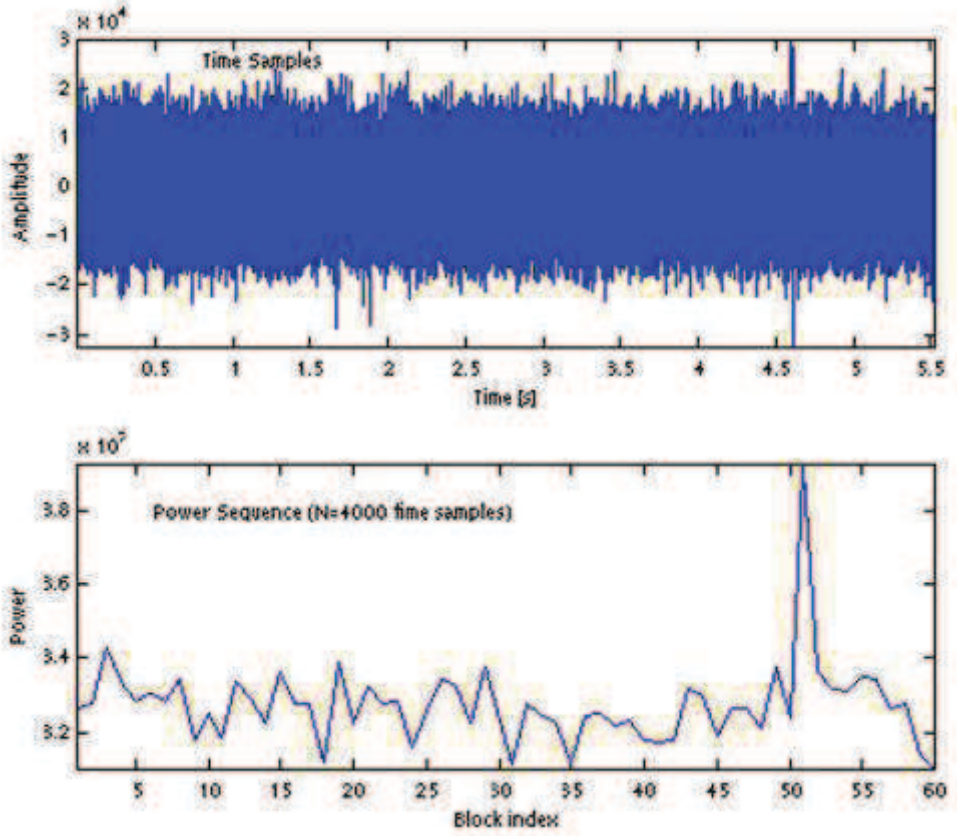


Рисунок 4. Аудиосигнал с выстрелом, а также его набор значений мощностей

В качестве примера приведен аудиосигнал с выстрелом, произошедшем на отметке 4.6 с, а также набор значений мощностей для блоков из N=4000 отсчетов, что соответствует продолжительности каждого блока около 90мс при частоте дискретизации сигнала 44100 Гц.

Различные методы отличаются способом автоматического детектирования блока, соответствующего резкому импульсному звуку. Можно выделить следующие группы способов:

1. Способы на основе стандартного отклонения нормированных значений мощностей блоков;
2. Способы на основе применения медианного фильтра для значений мощностей блоков;
3. Способы детектирования по динамическому порогу для значений мощностей блоков.

1.1.1 Метод детектирования аудиособытий на основе стандартного отклонения нормированных значений мощностей блоков

Ключевым аспектом данного метода является нормировка рассматриваемого набора значений блоков мощности к диапазону [0, 1]:

$$e_{norm}(j) = \frac{e_{win}(j) - \min\limits_{j}(e_{win}(j))}{\max\limits_{j}\left(e_{win}(j) - \min\limits_{j}(e_{win}(j))\right)}$$

Далее вычисляется стандартное отклонение (дисперсия) полученного набора значений:

$$var(k) = \frac{1}{L-1}\sum_{j=0}^{L-2}\left[e_{norm}(j,k) - \overline{e}_{norm}(k)\right]^2$$

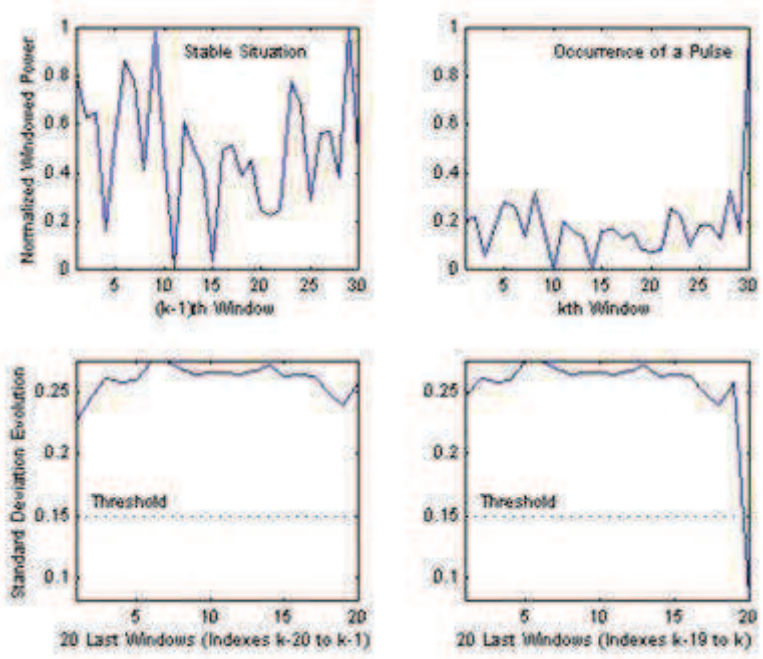


Рисунок 5. Изменение дисперсии мощности в разных блоках (с аудиособытием и без)

В случае фонового шума, значения блоков мощности будут примерно равномерно распределены в диапазоне [0-1] (видно на рисунке слева). Так как при поступлении нового значения мощности для блока аудиосигнала, происходит перенормировка значений к указанному диапазону, то при наступлении блока со значительно более высоким уровнем мощности, значение стандартного отклонения существенно уменьшится по сравнению с аналогичным значением для набора предыдущих фоновых блоков мощности. По снижению стандартного отклонения ниже порогового значения, можно автоматически детектировать блок с импульсным сигналом.

Преимуществом данного метода является его **устойчивость к изменению уровня шума, а также возможность детектирования медленно меняющегося сигнала**, анализируя среднее значение нормированных блоков мощности.

1.1.2 Метод детектирования аудиособытий на основе использования медианного фильтра

Основные этапы автоматического детектирования импульсных выделяющихся сигналов с использованием медианного фильтра представлены ниже. Пример применения медианного фильтра порядка k к

набору значений блоков мощностей представлен на рисунке ниже.

$$mf(k) = MED^{k}_{i=k-L+1}\ e(i)$$

Для детектирования блока с импульсным событием применяется условный медианный фильтр (conditional median filtering), который оставляет исходное значение сигнала в случае, если разница между исходным отсчетом и медианным значением меньше порогового значения, и медианное значение в противном случае.

$$cmf(k) = \begin{cases} mf(k) & if\ \left| mf(k) - e(k-d) \right| > th \\ e(k-d) & otherwise \end{cases}$$

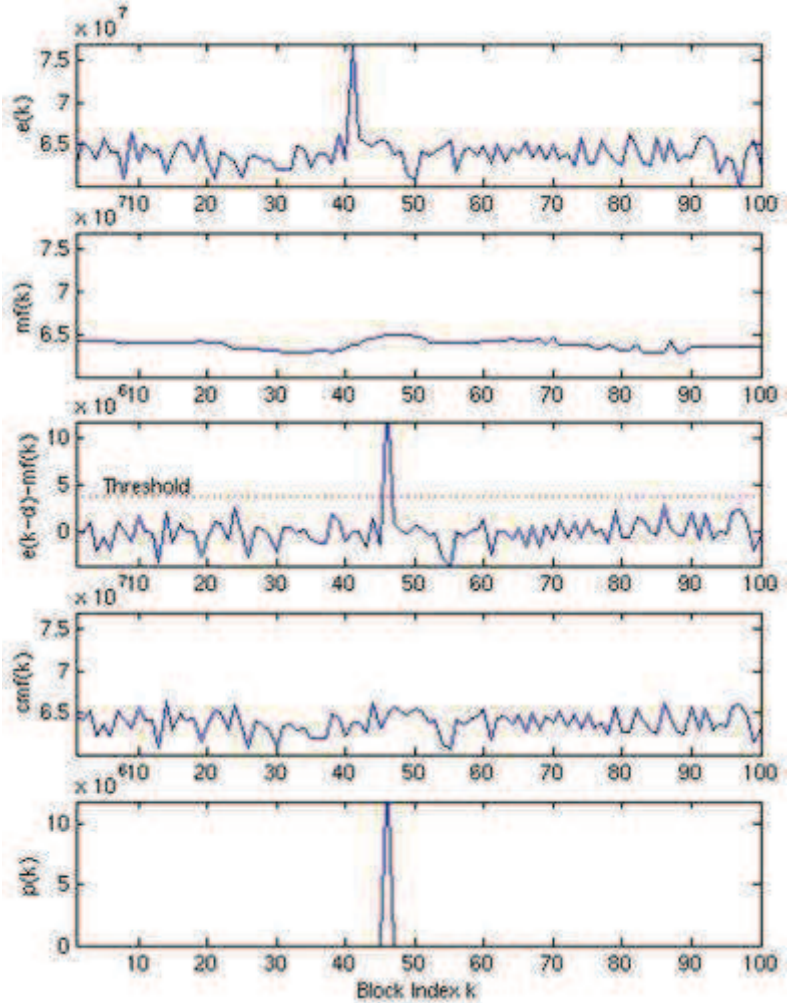


Рисунок 6. Этапы обработки сигнала с использованием медианного фильтра

Вычислив разницу между сигналом после применения *conditional median filter* и смещенным исходным сигналом, можно автоматически выделить блок с произошедшим импульсным событием.

1.1.3. Метод детектирования аудиособытия на основе динамического порога для значений мощностей блоков

В этом методе предлагается детектировать импульсный сигнал используя среднее значение набора мощностей блоков и среднеквадратичное отклонение в качестве динамического порога. Автоматическое детектирование происходит при превышении мощности очередного блока порогового значения, определяемого как

th=par*std+m

Где par – параметр, определяющий чувствительность алгоритма. Пример применения данного метода при значении параметра par=3 представлен на рисунке 7:

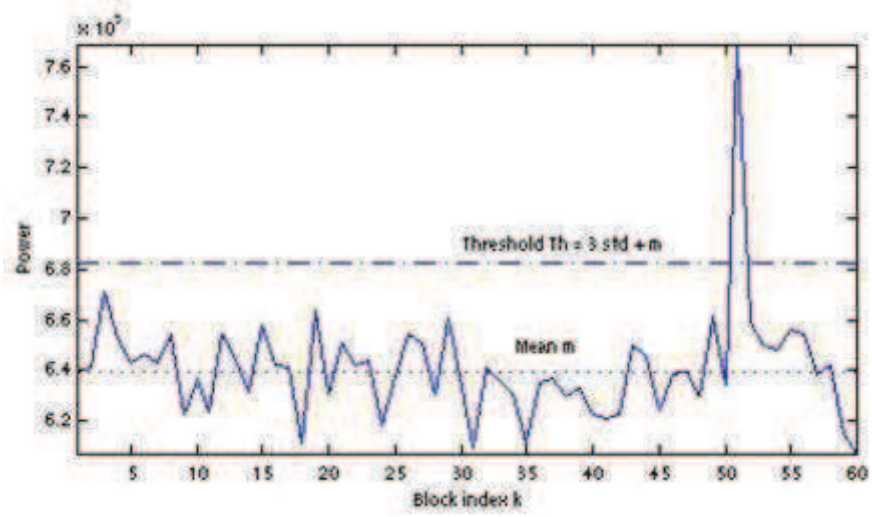


Рисунок 6. Пример обработки сигнала методом на основе динамического порога

1.2 Классификация аудиособытий

Следующим этапом после выделения является классификация (распознавание) аудиособытий. Как показано на рисунке 7 распознавание аудиособытий включает следующие этапы:

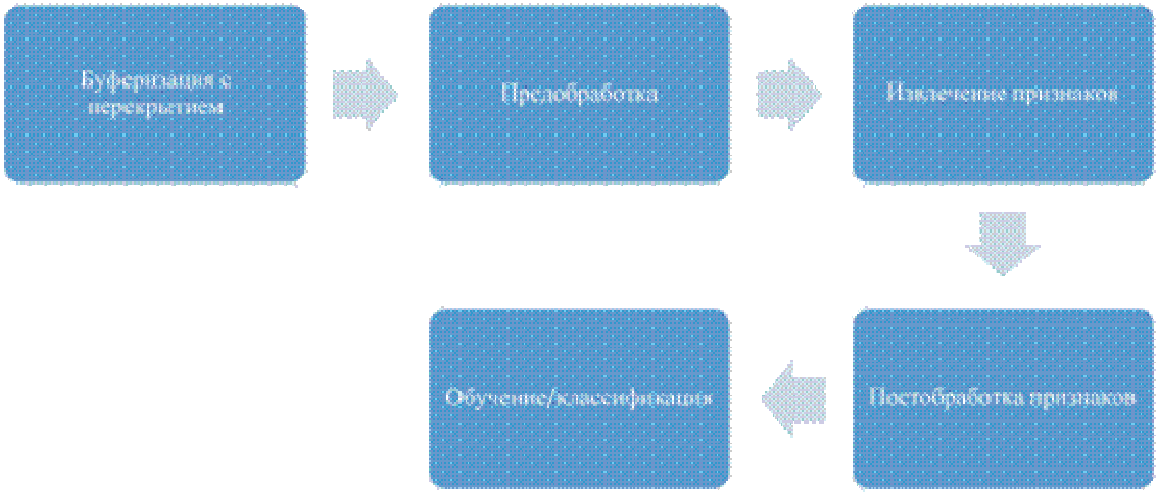


Рисунок 7. Общая схема распознавания аудиособытий

1.2.1 Буферизация с перекрытием

На первом этапе происходит преобразование исходного аудио сигнала в набор фреймов с перекрытием, т.е. конец одного фрейма должен пересекаться с началом другого.

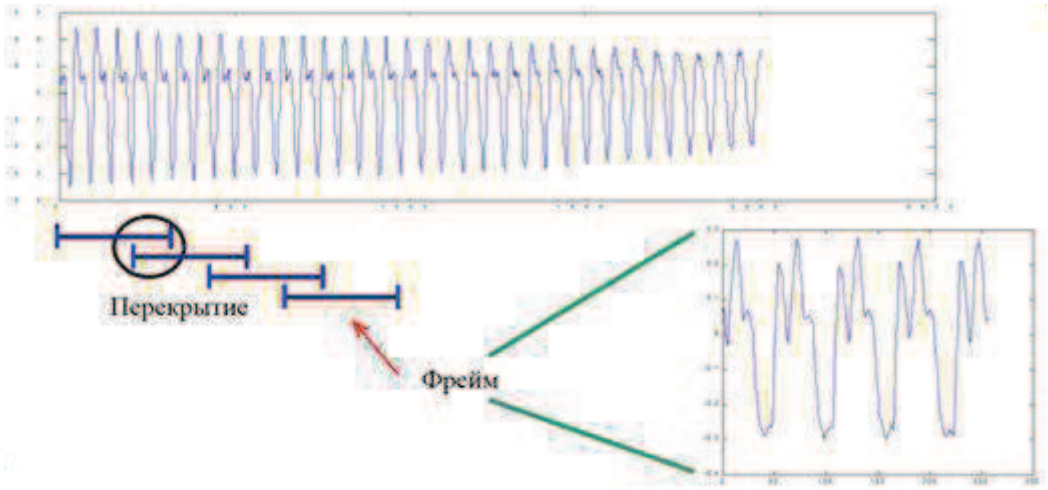


Рисунок 8. Буферизации с перекрытием

Фреймы являются более подходящей единицей анализа данных, чем конкретные значения сигнала, так как анализировать волны намного удобней на некотором промежутке, чем в конкретных точках. Расположение же фреймов “внахлёст” позволяет сгладить результаты анализа фреймов, превращая идею фреймов в некоторое “окно”, движущееся вдоль исходной функции (значений сигнала).

Опытным путём установлено, что оптимальная длина фрейма должна соответствовать промежутку в 10-100 мс, перекрытие — 50%. При частоте дискретизации трека 44100 Гц шаг в 100 мс будет соответствовать 4410 отсчетам.

1.2.2 Стадия предобработки

Стадия предобработки включает в себя, как правило, pre-emphasis фильтрацию и оконное взвешивание. Pre-emphasis обработка осуществляется за счет применения КИХ-фильтра $H(z) = 1 - a/Z$. Это необходимо для спектрального сглаживания сигнала. В таком случае сигнал становится менее восприимчивым к различным шумам, возникающим в процессе обработки.

Оконное взвешивание необходимо применять в связи с тем, что аудиофрейм ограничен во времени, поэтому при переходе в частотную область будет происходить эффект просачивания спектра боковых лепестков, связанное с формой спектра функции прямоугольного окна (он имеет вид $\sin(x) / x$). Поэтому, чтобы уменьшить влияние этого эффекта, применяется взвешивание исходного сигнала различного вида окнами, с формой, отличной от прямоугольной. Отсчеты входной последовательности умножаются на соответствующую функцию окна, что влечет за собой обнуление значений сигнала на краях выборки. В качестве взвешенных функций чаще всего выступают окна Хэмминга, Блэкмэна, плоское, Кайзеля-Бесселя, Дольфа-Чебышева. Спектральные характеристики некоторых приведены на рисунке 9.

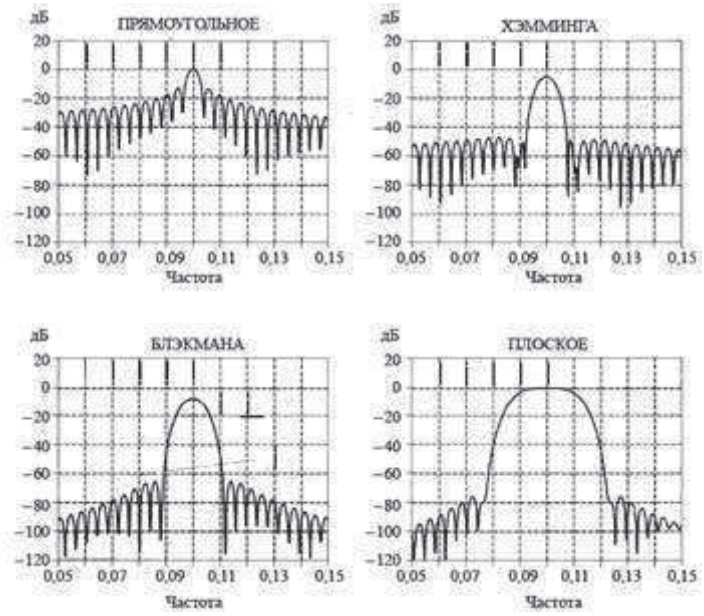


Рисунок 9. Некоторые Оконные функции

1.2.3 Извлечение признаков

Существует несколько подходов к извлечению дескрипторов (признаков) из аудиосигнала. Все они задаются общей целью уменьшить избыточность сигнала и выделить наиболее релевантную информацию, и, в то же время, отбросить нерелевантную. Как правило, признаки, описывающие аудиосигнал с разных точек зрения, комбинируются в один вектор признаков, на основе которого происходит процесс обучения и затем классификации с использованием выбранной обученной модели. Далее будут представлены наиболее популярные признаки, выделяемые из аудиосигнала.

1. Статистика во временной области (Time-domain statistics)

- ZCR (Zero crossing rate) – количество пересечений оси времени аудио сигналом.

$$zcr = \frac{1}{T-1}\sum_{t=1}^{T-1}\mathbb{I}\{s_t s_{t-1} < 0\}$$

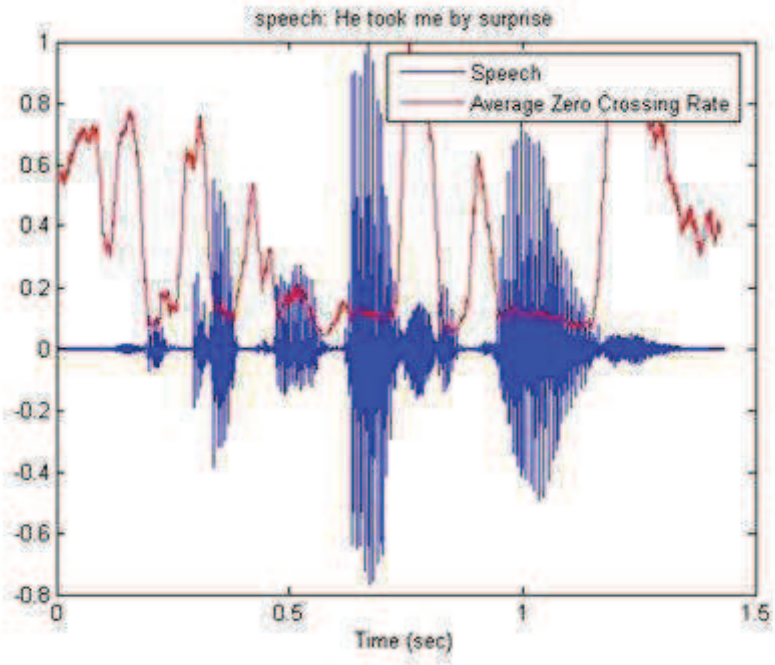


Рисунок 10. Характеристика Zero crossing rate

- Short-time energy – среднее значение энергии для аудио фрейма.

$$E_n = \frac{1}{T}\sum_{i=1}^{T} x_i^2$$

- Entropy of energy

Разделив каждый фрейм на набор подфреймов, вычисляется набор энергий для каждого подфрейма. Далее, нормируя энергию каждого из подфреймов на энергию всего фрейма, можно рассматривать набор энергий как набор вероятностей и вычислить информационную энтропию по формуле:

$$E_n = -\sum_i x_i * log2(x_i)$$

2. Статистика в частотной области (Frequency-domain statistic)

- Spectral centroid

Представляет собой интерпретацию «центра масс» спектра. Вычисляется как сумма частот, взвешенных соответствующими амплитудами спектра, деленная на сумму амплитуд:

$$Spectral\ \ Centroid\ = \frac{\sum_{k=1}^{N} kF[k]}{\sum_{k=1}^{N} F[k]}$$

Где F[k] – амплитуда спектра, соответствующая k-му значению частоты в спектре дискретного преобразования Фурье (ДПФ).

Затем полученное значение удобно отнормировать на максимальное значение частоты (Fs/2), в результате диапазон возможных «центр масс» спектра будет лежать в диапазоне [0-1].

- *Spectrum spread* (также называемый мгновенной шириной спектра/полосой пропускания).

Определяется как второй центральный момент:

$$SS = \frac{\sum_{k} (f_k - SC)^2 F[f_k]}{\sum_{k} F[f_k]}$$

Где fk – значения частот в ДПФ, F[fk] – значения амплитуд, SC – значение Spectral centroid

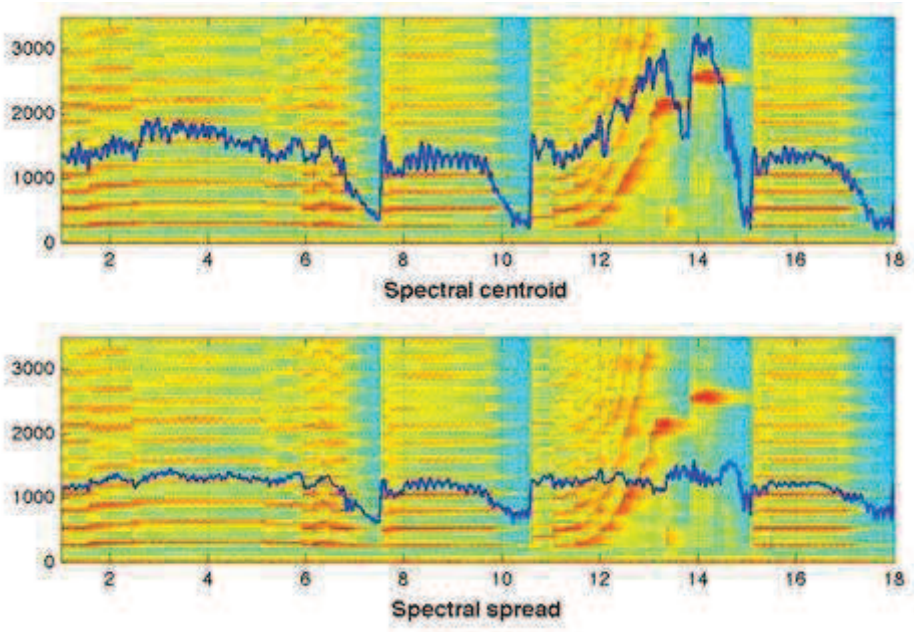


Рисунок 11. Характеристики Spectral centroid и Spectrum spread

- *Audio Spectrum Flatness (ASF)*

Отражает отклонение мощности спектра сигнала от пологой формы. С точки зрения человеческого восприятия, характеризует степень тональности звукового сигнала.

$$\text{Flatness} = \frac{\sqrt[N]{\prod_{n=0}^{N-1} x(n)}}{\frac{\sum_{n=0}^{N-1} x(n)}{N}} = \frac{\exp\left(\frac{1}{N}\sum_{n=0}^{N-1} \ln x(n)\right)}{\frac{1}{N}\sum_{n=0}^{N-1} x(n)}$$

- *Energy-band spectrum*

Пространство частот разделяется на N полос, после чего вычисляется энергия спектра в каждой полосе. Полученные значения берутся в качестве признаков. По сути признаки в таком случае представляют собой значение энергии спектра в «низком разрешении».

- *Cepstral coefficients*

Получаются на основе предыдущих признаков путем перевода их в кепстральное пространство. Кепстр представляет собой не что иное как «спектр логарифма спектра», вместо преобразования Фурье применяют дискретное косинусное преобразование (DCT):

$$c_i = \left[DCT\left(ln(a_i{}^2)\right)\right]^2$$

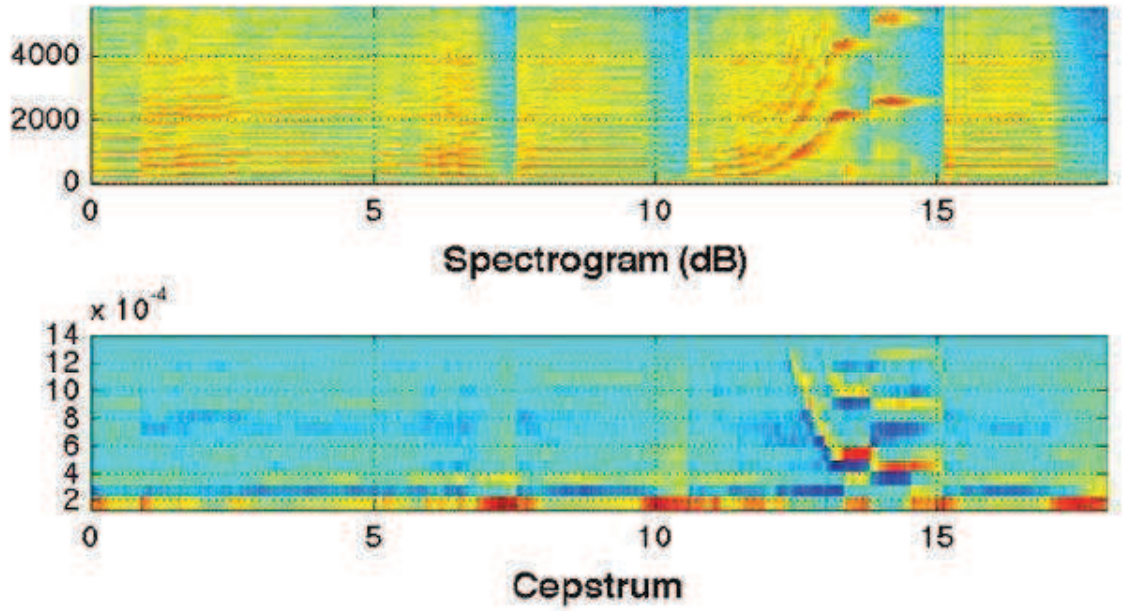


Рисунок 12. Характеристика - Кепстр

- *Spectral Entropy*

Энтропия спектра для заданного фрейма вычисляется аналогичным образом с энтропией энергии во временной области: частотная область спектра разделяется на N частотных подобластей, для каждой из которых рассчитывается часть всей энергии спектра, приходящаяся на данную подобласть, а затем вычисляется информационная энтропия по аналогии с временной областью.

- *Spectral flux*

Поток спектра отражает, насколько быстро изменяется энергия спектра, вычисляется на основе спектра на текущем и предыдущем фреймах. Определяется как вторая норма (Евклидово расстояние) между двумя нормализованными спектрами:

windowFFT = windowFFT / sum(windowFFT);

F = sum((windowFFT — windowFFTPrev).^2);

- *Spectral rolloff («завал» энергии)*

Определяется как относительная частота, в пределах которой которой сосредоточена определенная часть всей энергии спектра (задается в качестве параметра c):

countFFT -> c*totalEnergy;

SR = countFFT/lengthFFT;

Схематично представлено значение Spectral rolloff при c=0.95:

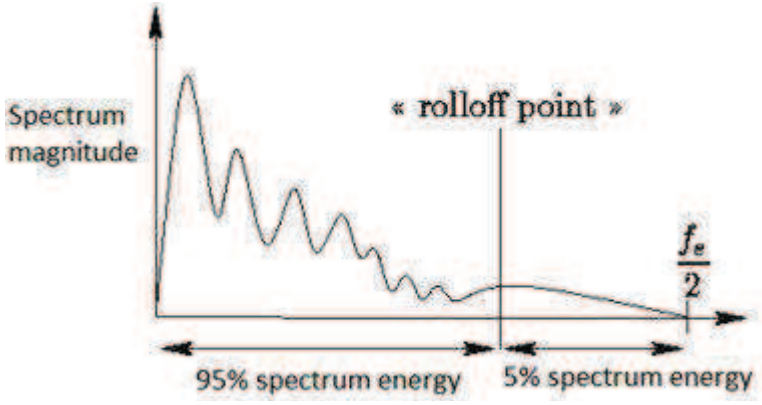


Рисунок 13. Характеристика - Завал энергии

1.2.4 Постобработка признаков

После извлечения необходимых признаков сигнала для их дальнейшего использования производится нормализация признаков так, чтобы каждый компонент вектора признаков имел среднее значение 0 и стандартное отклонение 1:

$$\bar{f}_k^{norm}(d) = \frac{\bar{f}_k(d) - \mu_d}{\sigma_d}$$

Часто применяется техника *mid-term analysis*, когда производится усреднение признаков по набору последовательных фреймов. Как правило, в качестве интервала для усреднения выбирается 1-10 секунд.

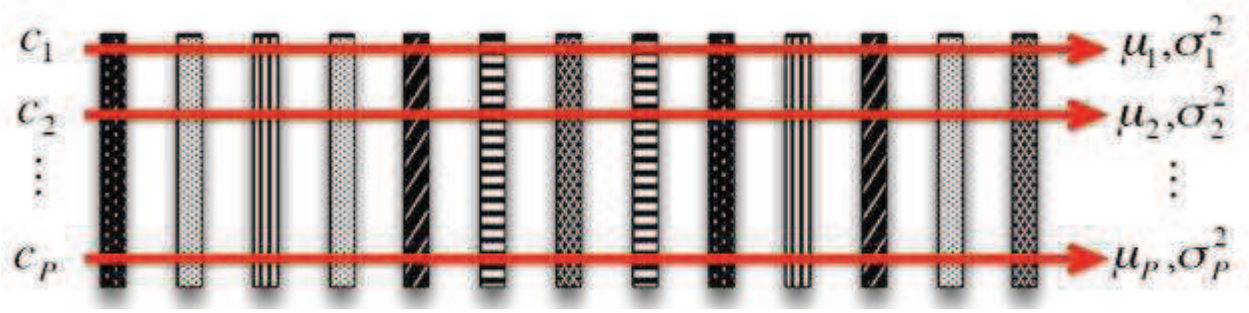


Рисунок 14. Нормализация признаков

Как правило, размерность вектора признаков получается довольно большой, что существенно влияет в дальнейшем на производительность процесса обучения. Вот почему в этом случае применяются хорошо известные и теоретически изученные методы сокращения размерности вектора признаков (LDA, PCA и др.).

Одним из преимуществ использования методов сокращения размерности является способность существенно увеличить скорость процесса

обучения за счет уменьшения количества признаков. Более того, выбрав признаки, обладающие наилучшими дискриминационными способностями в отдельный набор, возможно убрать лишнюю нерелевантную информацию, что повысит точность работы алгоритмов машинного обучения (таких как SVM и др.).

Данные методы были затронуты в материале про распознавание лиц. Различие PCA и LDA методов заключается в том, что метод PCA позволяет сократить размерность вектора признаков за счет выделения независимых компонент, максимальным образом покрывающий разброс по всем событиям.

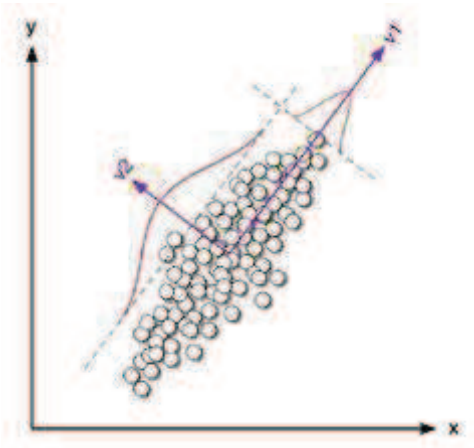


Рисунок 15. Метод PCA

В то время как LDA выделяет компоненты, показывающие наилучшие дискриминационные способности среди набора классов.

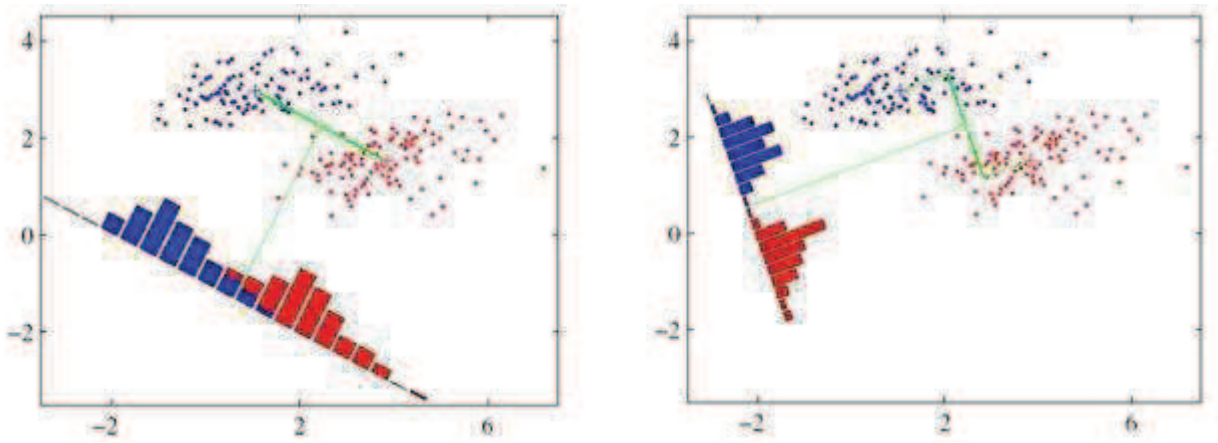


Рисунок 16. Метод LDA

Как в PCA, так и в LDA, на этапе обучения рассчитывается матрица W, определяющая линейное преобразование исходного вектора признаков в новое пространство сокращенной размерности Y = WTX.

1.2.5 Выбор классификатора

Наконец, последним этапом является выбор классификатора (обучающей модели). Некоторые исследования в области аудионалитики при детектировании интересующих аудиосигналов посвящены сравнению точности распознавания при использовании различных моделей классификаторов при различных типах аудиособытий. Отмечается, что использование иерархических классификаторов существенно увеличивает точность распознавания в сравнении с использованием мультиклассовых классификаторов.

Наиболее простым вариантом модели распознавания является **Байесовский классификатор**, который основан на вычислении функции правдоподобия для каждого из классов, и на этапе распознавания событие относится к тому классу, апостериорная вероятность которого максимальна среди всех классов. На следующем рисунке показан пример классификации для вектора признаков размерности 2. Эллипсы отцентрированы относительно средних значений, соответствующих каждому из классов, красные границы представляют собой набор значений признаков, где классы эквивалентны.

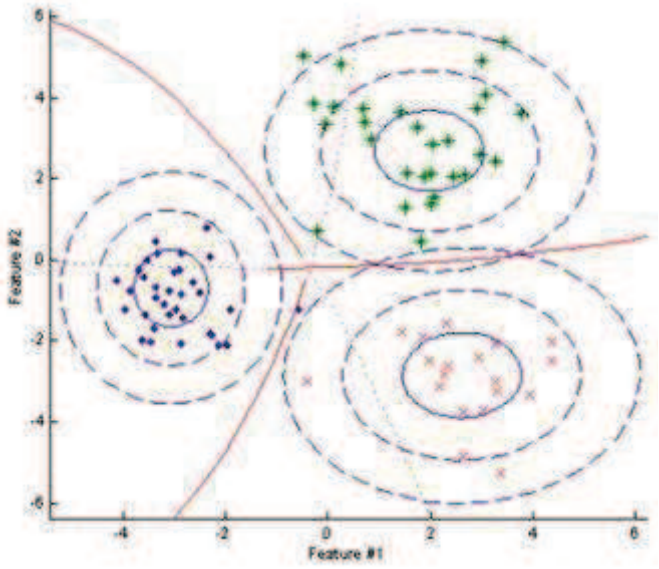


Рисунок 17. Байесовский классификатор

В случае выбора признаков, показывающих хорошие дискриминационные свойства, высокий результат распознавания может быть достигнут и с использованием байесовских классификаторов. В случае неудачного выбора признаков классификатор оказывается неэффективным, что относится к его **недостаткам**.

В отличие от классического Байесовского классификатора, где для аппроксимации каждого класса выступали параметры гауссовского распределения, в качестве функции плотности распределения вероятности в **GMM (Gaussian Mixture Model)** модели выступает «смесь» из нескольких гауссиан, параметры которых для каждого класса подбираются на этапе обучения с использованием EM (Expectation Maximization) алгоритма. Как видно из рисунка, представляющего результат классификации трех классов, область пространства, соответствующая каждому классу, представляет собой более сложную форму, чем эллипс, модель распознавания стала более точной, соответственно, улучшится результат распознавания.

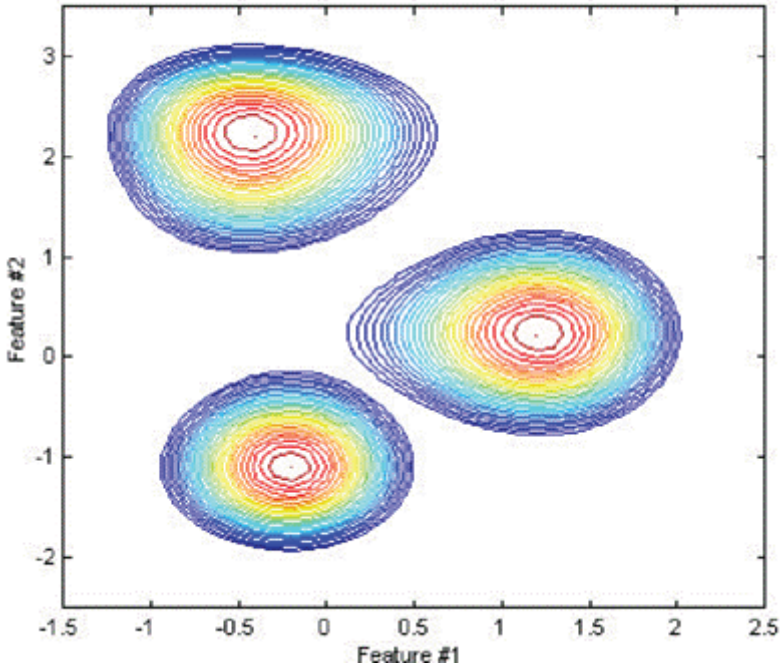


Рисунок 18. Gaussian Mixture Model

Среди **недостатков** данной модели можно отметить высокую чувствительность к вариациям в обучающей выборке данных при выборе большого числа гауссовских распределений, что может приводить к процессу переобучения.

В качестве еще одного способа классификации признаков, выделенных из аудиосигнала, является **SVM (Support Vector Machine)** классификатор, который переводит исходные вектора признаков в пространство с большей размерностью и находит максимально отстоящую от распознаваемых классов разделяющую гиперплоскость. В качестве ядра могут выступать не только линейная функция, но и полиномиальная, а также RBF (radial-basis function).

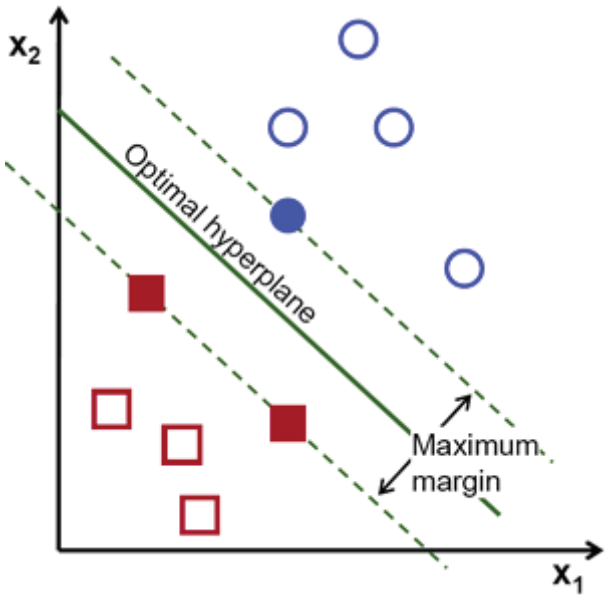


Рисунок 19. Support Vector Machine

Преимуществом данного классификатора в том, что на этапе обучения метод находит полосу максимальной ширина, что на этапе распознавания позволит осуществить более точную классификацию Среди **недостатков** можно выделить чувствительность к стандартизации данных и шумам.

Одним из статистических классификаторов являются **скрытые Марковские модели (СММ)** с дискретным временем. СММ используют статистические свойства сигналов и учитывают непосредственно их пространственные характеристики. Элементами модели являются: множество скрытых состояний, множество наблюдаемых состояний, матрица переходных вероятностей, начальная вероятность состояний. Каждому соответствует своя Марковская модель. При распознавании объекта проверяются сгенерированные для заданной базы объектов Марковские модели и ищется максимальная из наблюдаемых вероятность того, что последовательность наблюдений для данного объекта сгенерирована соответствующей моделью.

На сегодняшний день не удалось найти примера коммерческого применения СММ для распознавания звуков.

Недостатки СММ:

1. Необходимо подбирать параметры модели для каждой базы данных;
2. СММ не обладает различающей способностью, то есть алгоритм обучения только максимизирует отклик каждого образа на свою модель, но не минимизирует отклик на другие модели.

Также применяются в качестве классификатора **искусственные нейронные сети.**

ИНС является одним из самых популярных вариантов при выборе классификатора. Данная модель доказала свою эффективность в задачах распознавания изображений.

В настоящее время существует около десятка разновидности нейронных сетей (ИНС). Одним из самых широко используемых вариантов являться сеть, построенная на многослойном перцептроне, которая позволяет классифицировать поданное на вход изображение/сигнал в соответствии с предварительной настройкой/обучением сети.

Обучаются нейронные сети на наборе обучающих примеров. Суть обучения сводится к настройке весов межнейронных связей в процессе решения оптимизационной задачи методом градиентного спуска. В процессе обучения ИНС происходит автоматическое извлечение ключевых признаков, определение их важности и построение взаимосвязей между ними. Предполагается, что обученная ИНС сможет применить опыт, полученный в процессе обучения, на неизвестные образы за счет обобщающих способностей.

К недостаткам нейронных сетей можно отнести следующее:

1. Добавление нового эталонного лица в базу данных требует полного переобучения сети на всем имеющемся наборе (достаточно длительная процедура, в зависимости от размера выборки от 1 часа до нескольких дней);

2. Проблемы математического характера, связанные с обучением: попадание в локальный оптимум, выбор оптимального шага оптимизации, переобучение и т. д.;

3. Трудно формализуемый этап выбора архитектуры сети (количество нейронов, слоев, характер связей).

Обобщая все вышесказанное, можно заключить, что ИНС – «черный ящик» с трудно интерпретируемыми результатами работы, но при правильном подходе в обучении способна приносить хороший результат. Именно эта модель будет выбрана в качестве классификатора в данной работе.

1.3 Существующие модели поиска похожей музыки

1.3.1 Рекомендательная технология Диско

Компания "Яндекс" разработала технологию индивидуальных рекомендаций под названием Диско. Она используется в сервисах с крупными каталогами объектов — в Музыке, Радио, Маркете и Видео.

Диско в своем анализе использует информацию из нескольких источников. Во-первых, это поисковые запросы — они могут рассказать о текущих интересах. Во-вторых, это данные от технологии Крипта: пол, примерный возраст и род занятий. Они позволяют не рекомендовать человеку то, что ему заведомо не понравится. Скажем, 15-летней девочке, которая увлекается аквааэробикой, не стоит советовать музыку в жанре шансон. Наконец, это сведения от сервиса, для которого составляются рекомендации. Например, в Маркете это информация о том, какие товары просматривал человек, а в Музыке и Радио — какие треки он слушал.

Сигналы о предпочтениях пользователя могут быть положительными и отрицательными. Например, в Яндекс.Радио и Яндекс.Музыке композиции, которые пришлись не по душе, можно пропускать или отмечать оценкой «не нравится». Это отрицательный сигнал — он говорит о том, что в дальнейшем человеку такую музыку рекомендовать не надо. Кроме того, сигналы могут отличаться по весу. И оценка «мне нравится», и факт прослушивания трека от начала до конца являются положительными сигналами, но у первого вес будет больше.

Составляя рекомендации, Диско использует три разных подхода.

Первый подход опирается на информацию об объектах и связях между ними. Например, про любой музыкальный трек известно, на каком альбоме он вышел, кто его исполняет и к какому жанру он относится, а про любой товар — кто его производитель, каковы его характеристики и к какой категории товаров он принадлежит. Проанализировав связи, можно посоветовать пользователю объекты, родственные тому, чем он уже интересовался. Скажем, если человек часто слушает прогрессивный рок, ему можно предложить другие треки этого жанра, а если человек купил плиту и холодильник одного и того же производителя, а сейчас подбирает микроволновку, его, скорее всего, заинтересуют модели, выпущенные этой же компанией.

В основе второго подхода лежит информация о связях между людьми. Благодаря Крипте известны пол, примерный возраст и предположительный род занятий каждого пользователя. Установлено, что людей, у которых эти характеристики совпадают, часто интересуют одни и те же объекты. Даже если человек пользуется сервисом впервые и ещё не успел ничего посмотреть, послушать или приобрести, можно проверить, что смотрят, слушают или покупают люди со схожими характеристиками — и предложить ему эти же объекты.

Третий подход использует данные о взаимодействиях пользователей с объектами. Взаимодействием можно считать, например, факт просмотра видеоролика или оценку «нравится», поставленную музыкальному треку. Подход (в теории рекомендательных систем он известен как SVD — *singular value decomposition*, или сингулярное разложение) позволяет, опираясь на уже известные взаимодействия, предсказать, как пользователи отреагируют на те или иные объекты — например, какую оценку они поставят фильму, который пока не видели.

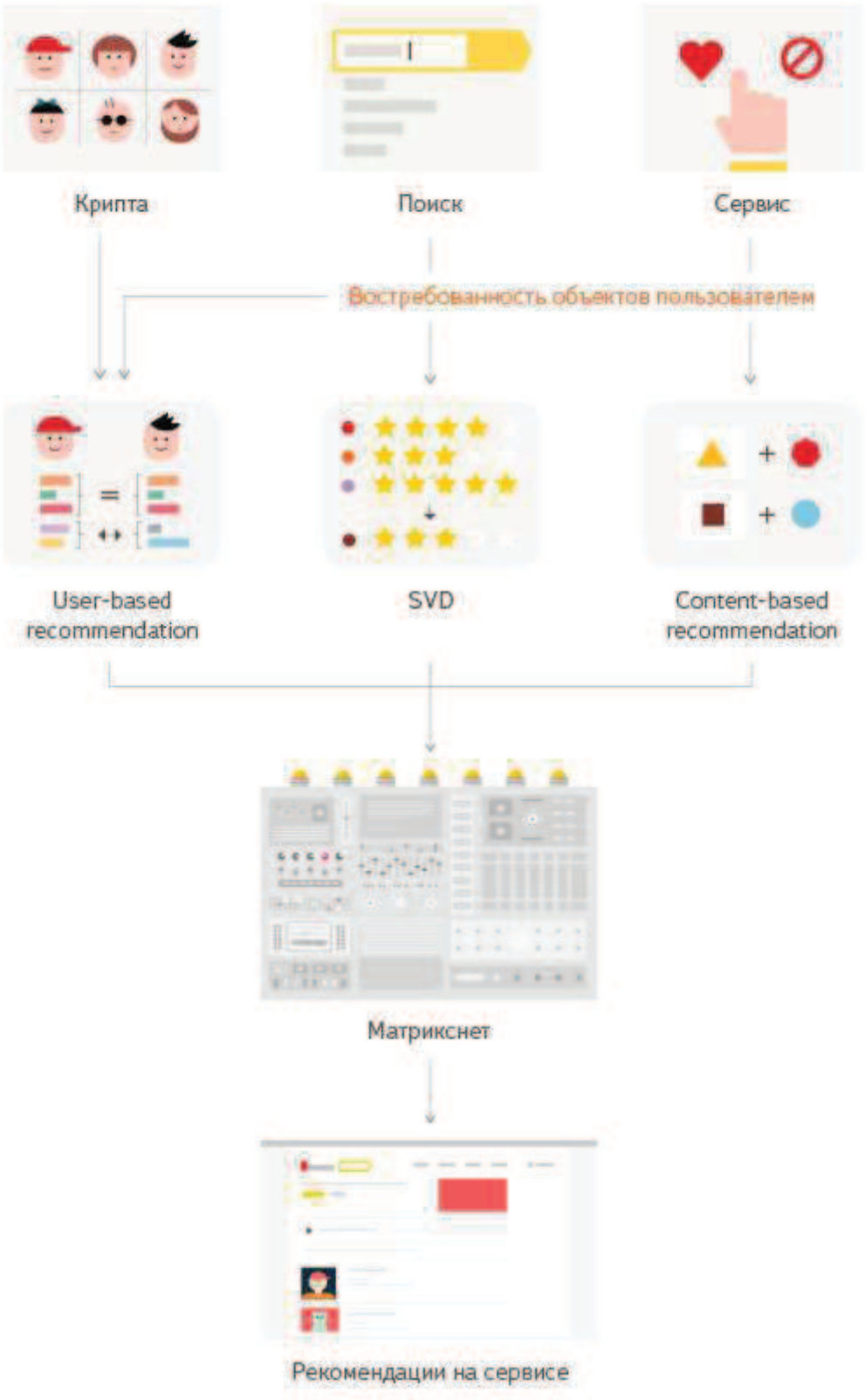


Рисунок 20. Технология Диско

У каждого из подходов есть свои достоинства. Первый подход позволяет посоветовать человеку редкие объекты, которыми мало кто

интересуется — например, малоизвестную музыкальную группу. Второй подход даёт возможность составлять рекомендации для людей, которые оказались на сервисе впервые и ещё не успели совершить никаких действий. Третий подход позволяет найти нетривиальные закономерности: скажем, может выясниться, что люди, которые интересуются надувными бассейнами и фитнес-трекерами, чаще других покупают кофеварки.

На этих трёх подходах основаны все рекомендательные модели, которые используются в Диско. Таких моделей насчитывается несколько сотен, и все они работают по-разному: одна составляет рекомендации с учётом музыкального жанра, вторая — с учётом бренда товаров, и так далее. Каждая модель на вход принимает набор параметров, а на выходе выдаёт список рекомендаций.

Все рекомендации от различных моделей обрабатывает метод машинного обучения Матрикснет. Его задача — составить сочетание рекомендаций, которое бы идеально соответствовало интересам пользователя в данный момент.

Таким образом, система Диско не использует в своем анализе непосредственно звуковой сигнал, что по мнению автора не составляет полной картины о звуковых предпочтениях пользователя.

1.3.2 Функция поиска похожего трека сервиса Яндекс.Музыка

Недавно сервис Яндекс.Музыка начал предлагать пользователям еще один способ открывать для себя новую музыку. Если, например, пользователь заслушали многократно слушал один из треков на сервисе, «Яндекс.Музыка» может предложить ему песню, и она будет похожа не только по жанру, но и по звучанию.

Для решения этой задач были использованы нейронные сети, обученные методом обучения с учителем. Нейросеть на основе спектрограммы трека в разные моменты времени понимает, например, что в какой-то момент в треке много высоких частот звука, а в другой момент, наоборот, преобладают низкие. И она начинает искать в спектрограмме другие такие зависимости. Это могут быть и не совсем понятные нам колебания звука, которые не факт, что действительно определяют наш запрос, а могут быть вполне очевидные вещи (например, смена ритма в середине песни). В итоге нейросеть переводит все эти особенности в цифры,

а сами треки получаются представлены относительно небольшим набором чисел (от нескольких десятков до пары тысяч). Математически это представление выглядит как вектор, и теперь нужно найти другие песни, чьи вектора будут похожи на вектор анализируемого трека.

Когда у нас есть представление трека в виде вектора, происходит этап выбора алгоритма сравнения. Здесь не обойтись без помощи человека. Сервис предлагаем людям послушать исходный трек и пару похожих, по мнению нейросетей, треков. А затем спрашиваем, какой трек из этой пары больше похож на исходный. После этого сервис можжет измерить, насколько точно решение алгоритма совпадает с оценкой людей и лучший из алгоритмов внедряется в «Яндекс.Музыку».

Хочется отметить, что данная система подбора находится на стадии исследований. Кроме того, поиск ведется по одному конкретному треку, что не дает возможности каким-то образом обобщить музыкальную коллекцию пользователя и вывести законномерности.

1.3.3 Система подбора похожего контента Genius

Рекомендательная система Genius - это продукт компании Apple. Отправной точкой для сервиса является пользовательские данные (песни, хранящиеся в библиотеке iTunes, и частота их исполнения), которые отправляются на сервера Apple, где они сливаются в одну большую базу данных всех пользователей службы.

Итак, Genius собирает информацию у вас на компьютере и объединяет ее с данными других пользователей. Служба сравнивает то, что находится внутри вашей коллекции с содержимым коллекций других пользователей, а затем обрабатывает полученную информацию с помощью секретного набора алгоритмов (похожие алгоритмы используются сервисами вроде Netflix).

Эта статистика считается глобально через регулярные промежутки времени и сохраняется в кеше. Данные о сходстве песен меняются редко – обычно с выходом новых релизов или изменением вкусов публики. В анализе используются алгоритмы поиска информации, в частности те, которые используют модель векторного пространства. При этом, расставляются приоритеты так, чтобы уделить наибольшее внимание важным факторам.

Важно отметить, что в процессе анализа используется статистическая мера. Она используется для оценки важности песен в контексте библиотеки, являющейся частью единой базы данных. (Вес отдельного трека пропорционален количеству воспроизведений этого трека в библиотеке, и

обратно пропорционален его частоте вопроизведения в других библиотеках всей базы данных на серверах Genius).

Другими словами, это механизм поиска похожих коллекций песен и плейлистов, которыми вы бы могли поделиться с другими пользователями, в попытке определить какое-то сходство. После того, как такая информация определена, Genius знает какую песню вам порекомендовать.

Кроме того, Genius также использует алгоритмы латентного фактора, которые как правило, хорошо работают с огромными массивами данных с большим числом измерений.

Получить более подробную информацию не представляется возможным. Но можно выделить, что в анализе используется статистическая мера.

Статистическая мера, использовалась первоначально для оценки важности слова в контексте документа, являющегося частью коллекции документов или корпуса. Вес некоторого слова пропорционален количеству употребления этого слова в документе, и обратно пропорционален частоте употребления слова в других документах коллекции. Статистическая мера часто используется в задачах анализа текстов и информационного поиска, например, как один из критериев релевантности документа поисковому запросу, при расчёте меры близости документов при кластеризации. Исходя из этого можно сделать предположение, что система ищет похожие музыкальные коллекции разных пользователей и рекомендует недостающие экземпляры их владельцам. Можно предположить, что данная система не использует в своем анализе звуковые сигналы.

1.4 Исследования в области неврологии

Исследователи под руководством Роберта Заторре из Института неврологии Монреаля провели следующий эксперимент. Девятнадцати добровольцам в возрасте от 18 до 37 лет (10 женщин, 9 мужчин), которые заранее сообщили о своих музыкальных вкусах, предложили прослушать и оценить 60 музыкальных треков. Важным условием было то, что испытуемые слушали эти произведения впервые. Участники эксперимента должны были оценить понравившиеся им треки по принципу аукциона, заплатив за них из собственных средств, некоторую сумму - 0.99, 1.29 или 2 доллара, с тем, чтобы по окончании эксперимента получить диск с выбранными музыкальными композициями. Во время всего эксперимента каждый участник был подключен к аппарату функционального МРТ, так что экспериментаторы могли видеть, как мозг реагирует на конкретное музыкальное произведение. И хотя продолжительность треков составляла всего 30 секунд, этого оказалось достаточно, чтобы мозг определил, нравится ему музыка, или нет. В ответ на понравившуюся музыку в мозгу

активировалось несколько зон, но самым чувствительным оказалось прилежащее ядро – область, которая активизируется, когда что-либо оправдывает наши ожидания. Оно входит в «центр удовольствия» головного мозга и проявляет активность при наркотическом и алкогольном опьянении и при половом возбуждении.

Этот эффект знаком каждому, ведь выбирая книгу или фильм мы, как правило, быстро оцениваем, нравится нам это или нет буквально по паре страниц или нескольким минутам экранного времени. Наш мозг способен предугадывать ощущения, опираясь на имеющиеся данные: если термометр показывает -10, значит, на улице холодно. Так же обстоят дела и с абстрактными эстетическими ожиданиями. Однако часто такие предсказания опираются на предыдущий опыт, а потому любитель рока, скорее всего, заскучает под народные напевы. Но в том случае, если впервые услышанная мелодия оправдывает ожидания, вырабатывается дофамин, который и приносит чувство удовольствия.

«Удивительно то, что человек предвкушает и возбуждается из-за чего-то совершенно абстрактного - из-за звука, который он должен услышать. У каждого человека прилежащее ядро имеет индивидуальную форму, из-за чего работает по-особенному. Также, стоит отметить, что из-за постоянных взаимодействий отделов мозга, с каждой мелодией у нас возникают собственные эмоциональные ассоциации», – прокомментировала результаты эксперимента, опубликованные в журнале «Science», доктор Валори Салимпур, одна из авторов исследования.

Прилежащее ядро связано и с другими областями мозга, а в случае со звуками задействована еще и слуховая кора. И чем больше нам нравятся те звуки, что мы слышим, тем сильнее это взаимодействие, тем больше образуется новых нейронных связей, которые, как известно, и составляют основу наших когнитивных способностей. Но чтобы прогнозировать, какую именно мелодию предпочтет каждый конкретный человек, необходимо знать его музыкальные вкусы, за которые отвечает височная доля. Связь между ней и прилежащим ядром ученые намерены исследовать в ближайшее время.

«Это очень интересно, поскольку любая мелодия состоит из отдельных звуков, каждый из которых по отдельности не имеет никакой ценности и не приносит удовольствия. Но когда мы слышим комбинацию этих звуков, то есть музыку, отделы нашего мозга, отвечающие за распознавание образов, прогнозирование и эмоциональное восприятие, начинают взаимодействовать между собой, и мы получаем эстетическое наслаждение», — прокомментировал работу Роберт Затторе.

По результатам данного исследования можно сделать вывод, что у человека есть вполне четкое представление идеальной для него музыки, основанное на предыдущем опыте. Это хорошо коррелируется с идеей о том, что на основе анализа музыкальной коллекции пользователя и выявленных при этом закономерностях, можно прогнозировать реакцию человека на ту или иную музыку.

ГЛАВА 2. ПЛАНИРОВАНИЕ

2.1 Входные данные

В качестве входной информации будут использованы треки в формате **mp3**. Выбор данного формата обусловлен его популярностью.

MP3 является одним из самых распространённых и популярных форматов цифрового кодирования звуковой информации с потерями. Он широко используется в файлообменных сетях для оценочной передачи музыкальных произведений. Формат может проигрываться практически во всех популярных операционных системах, на большинстве портативных аудиоплееров, а также поддерживается всеми современными моделями музыкальных центров и DVD-плееров.

MP3-файл состоит из нескольких фрагментов (фреймов) MP3, которые, в свою очередь, состоят из заголовка и блока данных. Такая последовательность фрагментов называется элементарным потоком. Фрагменты не являются независимыми элементами («резервуар байт»), и поэтому не могут быть извлечены произвольно. Блок данных MP3-файла содержит сжатую аудиоинформацию в виде частот и амплитуд. На рисунке 21 показано, что заголовок MP3 состоит из блоков, который служит для нахождения верного MP3-фрагмента. За ним следует бит, показывающий, что используется стандарт MPEG, и два бита, показывающие использование layer 3; другими словами, это определяет MPEG-1 Audio Layer 3 или MP3. Последующие значения могут варьироваться в зависимости от типа MP3-файла. Стандарт ISO/IEC 11172-3 определяет диапазон значений для каждой секции заголовка, вместе с общей его спецификацией. Большинство MP3-файлов в настоящий момент содержат ID3-метаданные, которые предшествуют или следуют за MP3-фрагментом; они также отображены на рисунке 21.

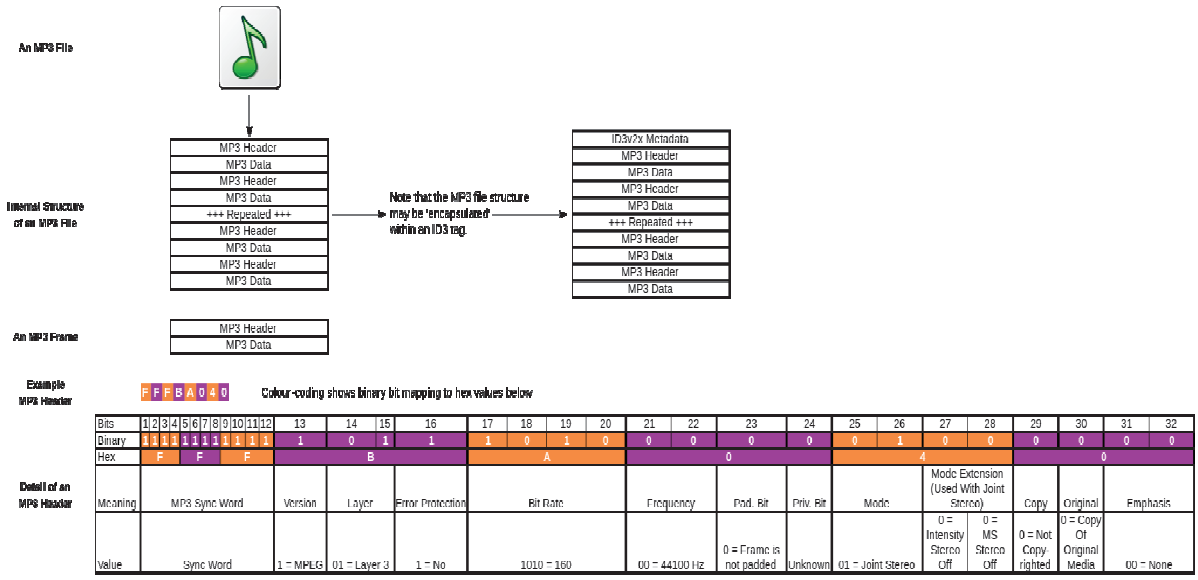


Рисунок 21. Структура mp3 файла

Треки каждого пользователя должны быть собраны в отдельной директории. Эта директория должна быть выбрана пользователем в начале работы программы.

2.2 Обработка аудиофайла

По результатам обзора решений и методов в области аудиоаналитики, проведенного в Главе 1 можно сделать следующие выводы:

1. В разрабатываемой системе при анализе следует использовать такую характеристику, как мощность сигнала или ее составляющую - магнитуду;

2. В разрабатываемой системе в качестве характеристик для описания аудио сигнала следует использовать: медиану, среднее, дисперсию и центральные моменты. Данные статистики нашли активное применение в области анализа аудиосигналов;

3. В разрабатываемой системе на первом этапе необходимо производить преобразование исходного аудио сигнала в набор фреймов длинной 100 мс, что соответствует 4410 отсчетам при частоте дискретизации 44100 Гц с перекрытием 50%;

4. В разрабатываемой системе необходимо применять оконные функции, чтобы уменьшить эффект просачивания спектра боковых лепестков;

5. В разрабатываемой системе следует использовать набор признаков, описывающих аудиосигнал с разных точек зрения. Данные признаки комбинируются в один вектор признаков, на основе которого происходит процесс обучения и затем классификации с использованием выбранной обученной модели.

6. При нормировке признаков в разрабатываемой системе следует применять технику *mid-term analysis*. Она заключается в усреднении признаков по набору последовательных фреймов. Как правило, в качестве интервала для усреднения выбирается 1-10 секунд.

7. В разрабатываемой системе следует ограничить количество классов, так как их увеличение отрицательно влияет на качество работы классификатора.

2.2.1 Разбиение на фреймы. Предобработка

После получения набора амплитуд во времени из mp3 аудиофайла следует разбить его но части (фреймы) по 4096 отсчетов с перекрытием 50 %. 4096 отсчетов при частоте дискретизации сигнала 44100 Гц соответствует временному интервалу в 93 мс. Число 4096 объясняется следующим этапом обработки.

Затем к каждому фрейму следует применить оконную функцию. В качестве нее выбрана функция Гаусса. Она позволит уменьшить эффект просачивания боковых лепестков, так как в дальнейшем при применении ДПФ будет получен дискретный набор частот.

Удачный FFT		**Неудачный FFT**	
Частоты	Коэффициенты (амплитуда)	Частоты	Коэффициенты (амплитуда)
95	0	91	0.04
96	0	93	0.11
97	0	96	0.14
98	0	99	0.85
99	0	102	0.19
100	1	105	0.15
101	0	108	0.09
102	0	111	0.02
103	0	114	0.001

Рисунок 22. Раскладывание на разные гармоники с помощью быстрого преобразования Фурье сигнала с частотой 100 Гц без применения оконных функций

Неудачный FFT		Неудачный FFT с Гаусса	
Частоты	Коэффициенты (амплитуда)	Частоты	Коэффициенты (амплитуда)
91	0.04	91	0.0011
93	0.11	93	0.013
96	0.14	96	0.021
99	0.85	99	0.93
102	0.19	102	0.14
105	0.15	105	0.011
108	0.09	108	0.003
111	0.02	111	0.0015
114	0.001	114	0.0001

Рисунок 23. Раскладывание на разные гармоники с помощью быстрого преобразования Фурье сигнала с частотой 100 Гц с применением оконных функций

Пример применения оконной функции Гаусса на языке C#:

```
double Gausse(double n, double frameSize)
{
    var a = (frameSize - 1)/2;
    var t = (n - a)/(Q*a);
    t = t*t;
    return Math.Exp(-t/2);
}
for (var i = 0; i < frameSize; i++)
{
    frame[i] *= Gausse(i, frameSize);
}
```

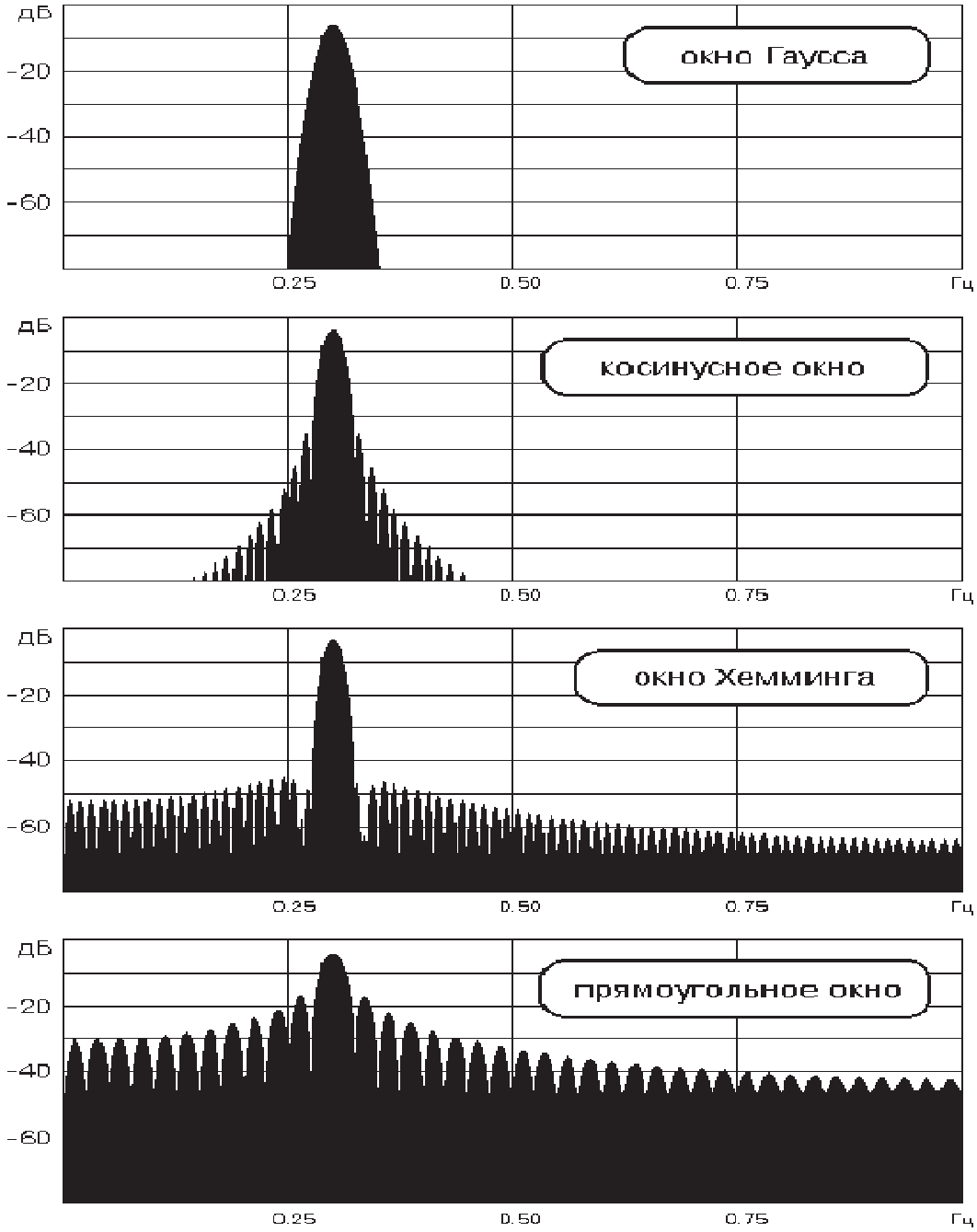


Рис. 12.7. Артефакты спектра одиночного гармонического сигнала (12.7) при использовании разных типов окно.

Рисунок 24. Артефакты спектра одиночного гармонического сигнала при использовании разных типов окон.

2.2.2 Преобразование Фурье

Для получения дискретного спектра сигнала следует использовать алгоритм **быстрого преобразования Фурье**, который минимизирует число математических операций, необходимых для его вычисления, на ЭВМ. Единственное требование алгоритма состоит в том, чтобы число отсчётов

было кратно степени двойки (256, 512, 1024 и так далее). Именно этим обусловлен выбор длины фрейма 4096, в пункте ранее.

Функция быстрого преоразования Фурье принимают массив комплексных чисел, заполненный реальными значениями амплитуд сигнала во временной области (мнимая часть у всех элементов равна 0), а после своего выполнения возвращают массив комплексных чисел, содержащий информацию об амплитудном и фазовом спектрах. Стоит отметить, что реальная и мнимая части комплексного числа — это далеко не то же самое, что его магнитуда и фаза. Их можно вычислить по следующим формулам (на примере языка C#):

```
magnitude = Math.Sqrt(x.Real*x.Real + x.Imaginary*x.Imaginary)
phase = Math.Atan2(x.Imaginary, x.Real)
```

В работе будет использована только магнитуда.

Результирующий массив комплексных чисел заполнен полезной информацией ровно на половину, другая половина является лишь зеркальным отражением первой и может быть исключена из рассмотрения. Этот момент объясняется теоремой Котельникова-Найквиста-Шеннона, о том, что частота дискретизации должна быть не меньше максимальной удвоенной частоты сигнала.

Сразу после вычисления преобразования Фурье необходимо нормализовать магнитудный спектр (пример на языке C#):

```
for (var i = 0; i < frameSize; i++)
{
        spectrum[i] /= frameSize;
}
```

Это приведёт к тому, что величина значений магнитуды получится одного порядка независимо от размеров фрейма.

Таким образом, будет получен массив длины 2048, каждый элемент которого будет представлять собой значение магнитуды сигнала на частоте кратной 10.76 Гц с учетом просачивания боковых лепестков. БПФ необходимо применить для каждой ранее выделенной во времени части сигнала. Данный вектор характеристик представляет собой аналогию с **энергией спектра в низком разрешении** (Energy-band spectrum).

2.2.3 Извлечение признаков. Уменьшение размерности

В результате, после действий в предыдущих пунктах получится последовательность частей сигнала во времени, разложенных в частотный спектр. Иными словами можно отследить для каждой частоты из дискретного набора, как менялось во времени ее присутствие (магнитуда).

Проблема заключается в том, что каждый фрейм имеет 2048 дискретных наборов частот. Для трека средней длины 180 секунд, получится около 1800 таких фреймов, что на выходе дает вектор длиной в районе 4 000 000. Это слишком много.

Для решения проблемы уменьшения размерности следует использовать элементы математической статистики.

Ранее были выделены: медиана, среднее, дисперсия, центральные моменты - , поэтому каждая частотная последовательность будет описана 5 характеристиками:

1) Среднее

2) Медиана

3) Дисперсия

4) Центральный момент 3 порядка (коэффициент асимметрии)

5) Центральный момент 4 порядка (коэффициент эксцесса)

Рассмотрим каждую характеристику отдельно:

- Среднее

  Среднее арифметическое (часто называемое просто средним) — наиболее распространенная оценка среднего значения распределения. Она является результатом деления суммы всех наблюдаемых числовых величин на их количество. Для выборки, состоящей из чисел $X_1, X_2, \ldots, X_n$, выборочное среднее (обозначаемое символом $\bar{X}$) равно $\bar{X} = (X_1 + X_2 + \ldots + X_n) / n$, *или*

$$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n}$$

  где $\bar{X}$ — выборочное среднее, n — объем выборки, X_i – i-й элемент выборки.

  Поскольку среднее арифметическое зависит от всех элементов выборки, наличие экстремальных значений значительно влияет на

результат. В таких ситуациях среднее арифметическое может исказить смысл числовых данных. Следовательно, описывая набор данных, содержащий экстремальные значения, необходимо наряду со средним арифметическим использовать медиану.

Применительно к данной задаче характеристика выступает в роле среднего значения энергии для отдельной полосы спектра сигнала.

- Медиана

Медиана представляет собой срединное значение упорядоченного массива чисел. Если массив не содержит повторяющихся чисел, то половина его элементов окажется меньше, а половина — больше медианы. Если выборка содержит экстремальные значения, для оценки среднего значения лучше использовать не среднее арифметическое, а медиану. Чтобы вычислить медиану выборки, ее сначала необходимо упорядочить.

$$\text{Медиана} = \frac{n+1}{2}\text{-й элемент упорядоченного массива}$$

Эта формула неоднозначна. Ее результат зависит от четности или нечетности числа n. Если выборка содержит нечетное количество элементов, медиана равна $(n+1)/2$-му элементу. Если же выборка содержит четное количество элементов, медиана лежит между двумя средними элементами выборки и равна среднему арифметическому, вычисленному по этим двум элементам.

Чтобы вычислить медиану выборки сначала необходимо упорядочить исходные данные.

Данная характеристика - медиана, нашла свое применение в **медианном фильтре**, используемом при детектировании сигнала.

- Дисперсия

Дисперсия позволяют оценить степень колебания данных вокруг среднего значения. **Выборочная дисперсия** является приближением среднего арифметического, вычисленного на основе квадратов разностей между каждым элементом выборки и выборочным средним.

Для выборки X_1, X_2, … X_n выборочная дисперсия (обозначаемая символом S^2 задается следующей формулой:

$$S^2 = \frac{(X_1 - \bar{X})^2 + (X_2 - \bar{X})^2 + ... + (X_n - \bar{X})^2}{n-1}$$

В общем случае выборочная дисперсия — это сумма квадратов разностей между элементами выборки и выборочным средним, деленная на величину, равную объему выборки минус один:

$$S^2 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1}$$

, где $\bar{X}$ — арифметическое среднее, n — объем выборки, X_i — i-й элемент выборки X.

Дисперсия позволяют оценить разброс данных вокруг среднего значения, иначе говоря, определить, сколько элементов выборки меньше среднего, а сколько — больше.

Дисперсия используется для определения **мгновенной ширины спектра** (Spectrum Spread).

- Коэффициент асимметрии

Для кривой нормального распределения характерно симметричное расположение отдельных значений относительно среднего, что можно проверить по величине **асимметрии**, которая является мерой косости.

$$K = \frac{\sum(X - \bar{X})^3}{n \cdot S^3}$$

где K - асимметрия;

S - среднее квадратическое отклонение;

X_i – текущее значение результатов испытаний;

$\bar{X}$ - среднее значение;

n - число испытаний.

K=0 свидетельствует о симметричности кривой распределения. Чем больше K, тем асимметричнее кривая.

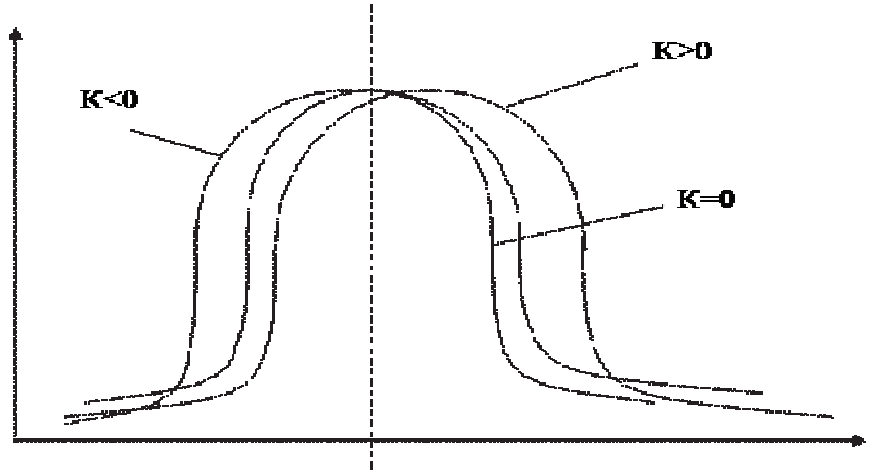


Рисунок 25. Форма кривой распределения с разными коэффициентами асимметрии

- Коэффициент эксцесса

Эксцесс (E) - позволяет судить о крутости кривой распределения по сравнению с кривой нормального распределения.

$$E = \left[\frac{\sum (X_i - \bar{X})^4}{nS^4} \right] - 3$$

Также используют коэффициент эксцесса без привязки его к закону нормального распределения. Для этого к формуле выше следует прибавить 3, что мы и будет использовано в данной работе.

Эксцесс характеризует относительную остроконечность или сглаженность распределения по сравнению с нормальным распределением. Положительный эксцесс обозначает относительно остроконечное распределение. Отрицательный эксцесс обозначает относительно сглаженное распределение.

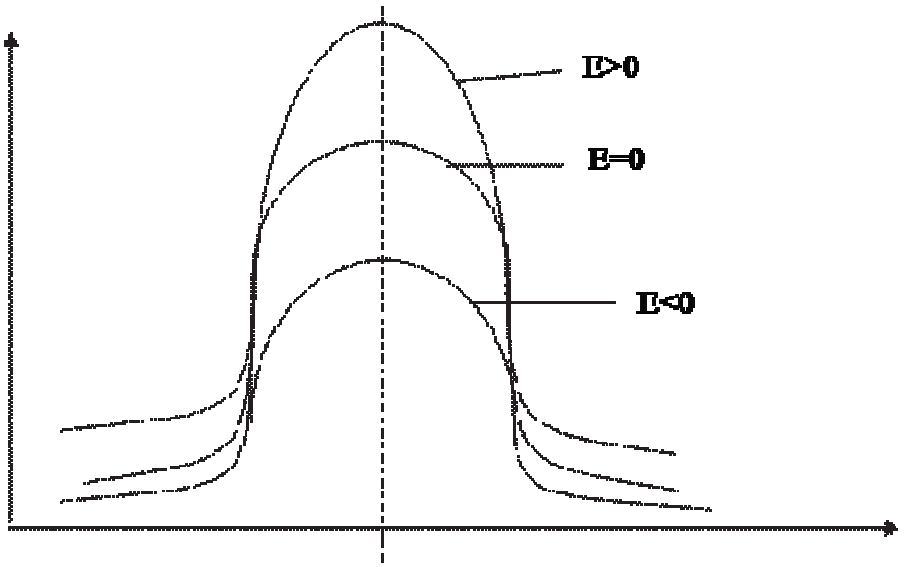


Рисунок 26. Форма кривой распределения с разными коэффициентами эксцесса

2.2.4 Формирование вектора признаков

В итоге, сигнал (аудиофайл) был разложен на наборе дискретных частот с шагом 10.76 Гц в последовательность магнитуды для каждой из частот во времени. На выходе получается вектор длины 10240 (2048 частот, по 5 характеристик на каждую), но данный вектор все равно является большим, чтобы подавать его на вход анализатора. Необходимо использовать один из методов уменьшения размерности.

В качестве метода уменьшения размерности вектора на данном этапе был использован метод **mid-term analysis**. Она заключается в усреднении признаков по набору последовательных фреймов. Усреднение следует производить для интервала 1 с, что соответствует последовательности из 10 характеристик. То есть происходить обьединение характеристик по 10 дискретным наборам, что соответствует полосе в 107.6 Гц. Усреднение следует производить по первым 2000 частот из набора.

В результате получен вектор фиксированной длины 1000. Данный вектор характеризует распределение магнитуды сигнала для 200 полос частот шириной около 100 Гц. Данная длина вектора является приемлемой для использования ее классификатором.

2.3 Классификатор

2.3.1 Выбор структуры ИНС

В качестве классификатора в данной работе была выбрана нейронная сеть. Нейронная сеть будет иметь архитектуру **Перцептрон с одним скрытым слоем**, что обусловлено ограниченными вычислительными возможностями. Входной слой будет иметь 1000 входов, что соответствует размерности вектора, полученного на этапе извлечения информации из аудиофайлов пользователей. Выходной слой должен иметь количество нейронов, соответствующее количеству пользователей в системе. В данной работе их будет 4. Выходной слой будет иметь тип softmax, что даст на выходе вероятности попадания в тот или иной класс. Количество нейронов в скрытом слое выбирается исходя из эмпирического правила для данной модели и соответствует полусумме входных и выходных нейронов в сети, а именно 500 нейронам. Активационной функцией для скрытого слоя выбрана функция гиперболического тангенса, вычисляемая по формуле:

$$Out = \frac{e^{NET} - e^{-NET}}{e^{NET} + e^{-NET}}$$

Функция часто применяется в сетях с непрерывными сигналами. Ее особенность в том, что она может возвращать отрицательные значения результата.

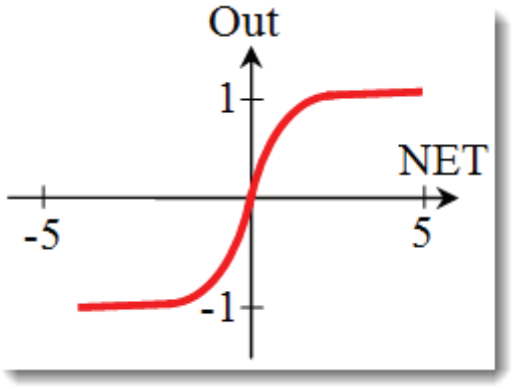


Рисунок 27. Функция гиперболического тангенса

Данная модель однослойного перцептрона с одним скрытым слоем активно применяется в задачах классификации. Модель однослойного перцептрона менее эффективна, чем модель перцептрона с несколькими скрытыми слоями, но выбор именно этой модели обьясняется условиями задачи. Во-первых, обучающая выборка содержит от 500 до 2500 примеров, что недостаточно для применения модели многослойного перцептрона,

которая подразумевает большое количество примеров. Во-вторых, меньшее количество скрытых слоев несколько компенсируется за счет вычисления характеристик сигнала на этапе формирования входного вектора. В-третьих, однослойный перцептрон обучается значительно быстрее, так как имеет меньшее количество связей. На основании этих факторов предпочтение было отдано модели с одним скрытым слоем.

2.3.2 Нормировка

Нормализация входных данных - это процесс, при котором все входные данные проходят процесс "выравнивания", т.е. приведения к интервалу [0,1] или [-1,1]. Если не провести нормализацию, то входные данные будут оказывать дополнительное влияние на нейрон, что приведет к неверным решениям, так как сеть будет учитывать только входы со значениями высоких порядков.

В общем виде формула нормализации выглядит так:

$$y = \frac{(x - x_{\min})(d2 - d1)}{x_{\max} - x_{\min}} + d1$$

где:

- x - значение, подлежащее нормализации;
- $[x_{\max}, x_{\min}]$ - интервал значений **x**;
- $[d1, d2]$ - интервал, к которому будет приведено значение **x**.

Перед тем как, начать обучение, следует найти максимальное и минимальное значение по всем входным векторам. Затем, используя формулу выше привести все значения во входных векторах к интервалу [0,1].

Также к нормировки относится фиксированный размер входного вектора и порядок размещения параметров в нем. Эти факторы учтены на этапе формирования входного вектора.

2.3.3 Обучение

Обучение перцептрона состоит в подстройке весовых коэффициентов каждого нейрона. Пусть имеется набор пар векторов (x^a, y^a), a = 1..p, называемый **обучающей выборкой**. Будем называть нейронную сеть обученной на данной обучающей выборке, если при подаче на входы сети каждого вектора x^a на выходах всякий раз получается соответсвующий вектор y^a.

Рассмотрим классический для модели сети перцептрон алгоритм для обучения нейронной сети - **алгоритм обратного распространения ошибки (BackProp)**.

Целью обучения сети алгоритмом обратного распространения ошибки является такая подстройка ее весов, чтобы приложение некоторого множества входов приводило к требуемому множеству выходов. При обучении предполагается, что для каждого входного вектора существует парный ему целевой вектор, задающий требуемый выход. Вместе они называются обучающей парой. Сеть обучается на многих парах.

Метод обучения обратного распространения ошибки состоит в итерационной подстройке матрицы весов, последовательно уменьшающей ошибку в выходных векторах. Алгоритм включает несколько шагов:

Шаг 0. Инициализировать синаптические веса маленькими случайными значениями.

Шаг 1. Выбрать очередную обучающую пару из обучающего множества; подать входной вектор на вход сети.

Шаг 2. Вычислить выход сети. Затем вычислить разность между выходом сети и требуемым выходом (целевым вектором обучающей пары).

Шаг 3. Подкорректировать веса сети для минимизации ошибки. Этот шаг повторяется слой за слоем по направлению к входу, пока все веса не будут подкорректированы.

Шаг 4. Шаги 1-3 повторяются для всех обучающих векторов. Один цикл последовательного предъявления всей выборки называется *эпохой*. Обучение завершается по истечении нескольких эпох, а) когда итерации сойдутся, т.е. вектор весов перестает изменяться, или б) когда полная просуммированная по всем векторам абсолютная ошибка станет меньше некоторого малого значения.

Корректировка весов сети производится по формуле:

$$w_{p\text{-}q}(i+1) = w_{p\text{-}q}(i) + \eta \delta_q OUT_p$$

где:

i – номер текущей итерации обучения;

$w_{p\text{-}q}$ – величина синаптического веса, соединяющего нейрон p с нейроном q ;

η (греческая буква «эта») – коэффициент «скорости обучения», позволяет управлять средней величиной изменения весов;

OUT_p – выход нейрона p.

Представленный алгоритм относится к широкому классу алгоритмов **обучения с учителем**, поскольку известны как входные вектора, так и требуемые значения выходных векторов (имеется учитель, способный оценить правильность ответа ученика).

Доказанная Розенблаттом теорема о сходимости обучения по d - правилу говорит о том, что персептрон способен обучится любому обучающему набору, который он способен представить.

В данной работе в качестве алгоритма обучения будет использована модификация алгоритма обратного распространения ошибки – алгоритм **Rprop** (Resilent Propogation – «упругое распространение»). В этом алгоритме устранен основной недостаток BackProp - скорость обучения. Rprop использует знаки частных производных для подстройки весовых коэффициентов.

В процессе обучения следует учесть следующие важные особенности:

1) Если данных мало (или они мало репрезентативные), сеть можно переобучить. Нейросеть максимально подстроится под выборку и потеряет работоспособность на реальных данных. Для того, чтобы контролировать обобщающие способности следует разделите все данные на три выборки соотношении 70: 20: 10. Обучаться следует на **Train**, периодически проверяя качество модели на **Test**. Для финальной непредвзятой оценки использовать **Validation**.

2) Следует применять технику регуляризации. **Регуляризация** – это техника, которая позволяет избежать переобучения нейросети во время обучения, даже если данных мало. Признак переобучившейся нейросети – большие значения весов, порядка сотен и тысяч, такая нейросеть не будет нормально работать на новых, не виденных ранее, данных. Это связано с тем, что при больших весах активационная функция нейрона входит в зону "насыщения" и имеет плохой отклик (ее вес практически не меняется).

3) Для искусственных нейронных сетей на современном этапе технического развития практика **дообучения** является рискованной: сеть может переобучиться или подстроиться под самые последние поступившие данные – и потеряет свои обобщающие способности.

Учтя все вышеизложенные особенности можно составить следующею последовательность действий при обучении:

1) Обучение;

2)Тестирование качества на тестовых (в процессе обучения) и валидационных выборках;

3) Выбор удачного варианта сети с фиксированием ее весов.

Использовать обученную нейросеть на практике, следует без изменения ее весов.

Глава 3. РАЗРАБОТКА

3.1 Термины и сокращения

Графический интерфейс поользователя (англ. graphical user interface, GUI) - разновидность пользовательского интерфейса, в котором элементы интерфейса (меню, кнопки, значки, списки и т. п.), представленные пользователю на дисплее, исполнены в виде графических изображений.

Амплитудно-частотная характеристика (АЧХ) — зависимость амплитуды выходного сигнала от частоты входного гармонического сигнала.

Быстрое преобразование Фурье (**БПФ**, *FFT*) — алгоритм быстрого вычисления дискретного преобразования Фурье (ДПФ). То есть, алгоритм вычисления за количество действий, меньшее чем , требуемых для прямого (по формуле) вычисления ДПФ. Иногда под БПФ понимается один из быстрых алгоритмов, называемый алгоритмом прореживания по частоте/времени или алгоритмом по основанию 2, имеющий сложность .

Частота дискретизации (или **частота семплирования**, англ. **sample rate**) — частота взятия отсчетов непрерывного во времени сигнала при его дискретизации (в частности, аналого-цифровым преобразователем). Измеряется в герцах.

BLOB (англ. *Binary Large Object* — двоичный большой объект) — массив двоичных данных. В СУБД BLOB — специальный тип данных, предназначенный, в первую очередь, для хранения изображений, а также компилированного программного кода.

Иску́сственная нейро́нная се́ть (ИНС) — математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма. Это понятие возникло при изучении процессов, протекающих в мозге, и при попытке смоделировать эти процессы.

ИНС представляют собой систему соединённых и взаимодействующих между собой простых процессоров (искусственных нейронов). Такие процессоры обычно довольно просты (особенно в сравнении с процессорами, используемыми в персональных компьютерах). Каждый процессор подобной сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим процессорам. И, тем не менее, будучи соединёнными в достаточно большую сеть с управляемым

взаимодействием, такие локально простые процессоры вместе способны выполнять довольно сложные задачи.

Искуственный нейрон - узел искусственной нейронной сети, являющийся упрощённой моделью естественного нейрона. Математически, искусственный нейрон обычно представляют как некоторую нелинейную функцию от единственного аргумента — линейной комбинации всех входных сигналов. Данную функцию называют функцией активации или функцией срабатывания, передаточной функцией. Полученный результат посылается на единственный выход.

Перцептрон - математическая или компьютерная модельвосприятия информации мозгом (кибернетическая модель мозга), предложенная Фрэнком Розенблаттом.

Обучение с учителем (Supervised learning) — один из разделов машинного обучения, посвященный решению следующей задачи. Имеется множество объектов (ситуаций) и множество возможных ответов (откликов, реакций). Существует некоторая зависимость между ответами и объектами, но она неизвестна. Известна только конечная совокупность прецедентов — пар «объект, ответ», называемая обучающей выборкой. На основе этих данных требуется восстановить зависимость, то есть построить алгоритм, способный для любого объекта выдать достаточно точный ответ. Для измерения точности ответов определённым образом вводится функционал качества.

Под учителем понимается либо сама обучающая выборка, либо тот, кто указал на заданных объектах правильные ответы. Существует также обучение без учителя, когда на объектах выборки ответы не задаются.

Mp3 - кодек третьего уровня, разработанный командой MPEG, лицензируемый формат файла для хранения аудиоинформации.

Систе́ма управле́ния ба́зами да́нных (СУБД) — совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

.NET Framework — программная платформа, выпущенная компанией Microsoft в 2002 году. Основой платформы является общеязыковая среда исполнения Common Language Runtime (CLR), которая

подходит для разных языков программирования. Функциональные возможности CLR доступны в любых языках программирования, использующих эту среду.

Интерпретатор – программа (разновидность транслятора), выполняющая интерпретацию.

Интерпретация — пооператорный (покомандный, построчный) анализ, обработка и тут же выполнение исходной программы или запроса (в отличие от компиляции, при которой программа транслируется без её выполнения).

3.2 Основные сведения

Наименование программы: «Анализатор музыкальных предпочтений».

Назначение и область применения: Программа предназначена для анализа аудиоданных пользователя без использования метаданных, подбора музыкальных композиций.

3.3 Требования к программе

3.3.1 Требования к GUI

Взаимодействие программы с пользователем должно происходить посредством графического интерфейса (GUI). При первом пуске программы должна быть выбрана директория с музыкальным контентом. В цикле работы с директорией должна присутствовать возможность ее изменения, с последующим дополнительным анализом. Элементы управления должны быть сгруппированы однотипно – горизонтально либо вертикально – на всех страницах интерфейса.

Интерфейс должен включать:

- Отображение композиций пользователей;

- Отображение информации о процессе анализа музыкального контента;

- Выдавать подборку рекомендуемых композиций (на базе композиций, добавленных пользователем для отбора).

3.3.2 Функциональные требования

Работа осуществляется только с файлами формата mp3. Пользователю будет предложено напрямую выставить оценки композициям или выставить их косвенно (путем сбора статистики прослушивание), что затянет процесс анализа, но сделает его более объективным.

Пользователю необходимо обеспечить количество треков в фонотеке количеством не менее 100. Меньшее количество может привести к нехватке данных для обучения нейронной сети.

Поиск материала для анализа осуществляет пользователь. Все это заносится в специальную директорию. Должна быть реализована возможность удаления неподходящей музыки после процедуры отбора из данной директории.

Процесс анализа не начнется до тех пор, пока вся музыка так или иначе не будет оценена пользователем.

3.4 Требования к видам обеспечения

3.4.1 Требования к хранению данных

Все данные должны храниться в структурированном виде под управлением реляционной СУБД. Исключения составляют файлы музыкальных треков. Такие файлы сохраняются в файловой системе, а в БД размещаются ссылки на них.

В качестве СУБД должна использоваться компактная встраиваемая открытая реляционная база данных SQLite. Приложение не потребует от пользователя наличие какой-либо СУБД. Возможностей SQLite вполне хватит для поставленных задач.

3.4.2 Требования к языкам программирования

Код должен быть написан с использованием .NET. Основной код программы должен быть выполнен на языке C#. Исключение составляет скрипты, написанные на IronPython, для связи с внешними модулями.

Для реализации Анализатора(нейронной сети) удобно использовать Python ввиду большого количества специализированных библиотек с полным набором удобным инструментов.

3.4.3 Требования к производительности

Приложение выполняет в процессе анализа достаточно затратные по ресурсам операции. Скорость работы будет существенно зависеть от мощности ПК пользователя. Код следует распараллелить для улучшения производительности на многоядерных компьютерах. Работа приложения нежелательна в фоновом режиме.

3.4.4 Требования к аудио файлам

Все файлы должны быть в формате mp3. Их количество, размеры и длительность ограничены лишь наличием свободного места на ПК пользователя. Файлы должны быть с частотой дискретизации 44100 Гц.

3.4.5 Требования к аппаратному обеспечению

Для функционирования программы необходимо следующее техническое обеспечение со следующими минимальными характеристиками:

- процессор – Intel Pentium III 2 Ghz;
- оперативная память – 512 Mb RAM;
- жесткий диск - 20 Gb HDD.

Желательно использование многоядерного процессора для ускорения процесса работы.

3.5 Проектирование

Для начала проектирования программной системы на основе требований составлена диаграмма использования, представленная на рисунке 28.

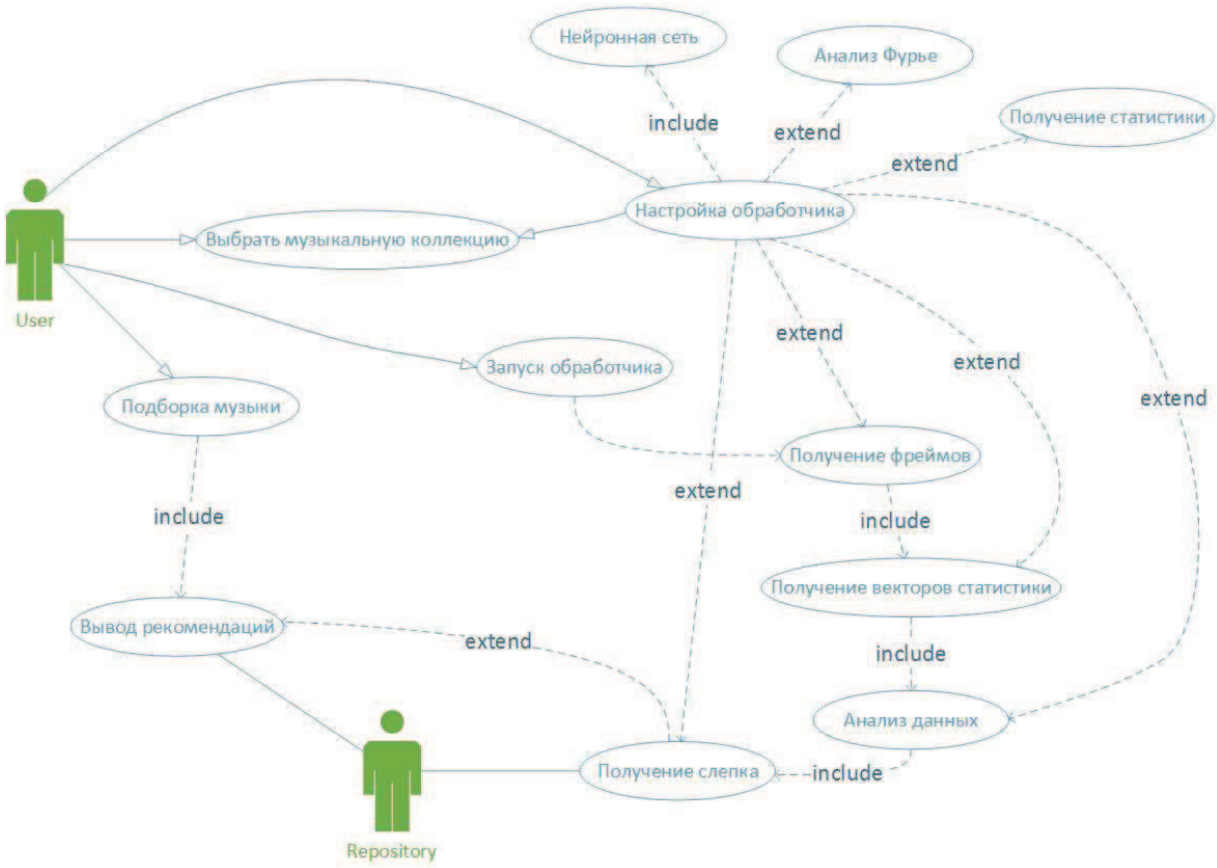


Рисунок 28. Диаграмма использования

Исходя из диаграммы использования создается объектная модель системы. Для этого необходимо представить работу анализатора в программных системах.

Понятие класс в объектно-ориентированном подходе означает некую группу объектов, описанных с необходимым уровнем абстракции, имеющие одинаковые свойства и связи с другими объектами. В терминах программирования класс есть абстракция или, еще это называют, типом данных.

Экземпляры классов в терминах программирования называются объектами, т.е. конкретными экземплярами того или иного типа данных (класса).

В языках программирования объектные свойства – это методы класса, через которые класс может взаимодействовать с другими классами. Свойства-значения же подходят под описание полей класса, в которых класс хранит те или иные данные.

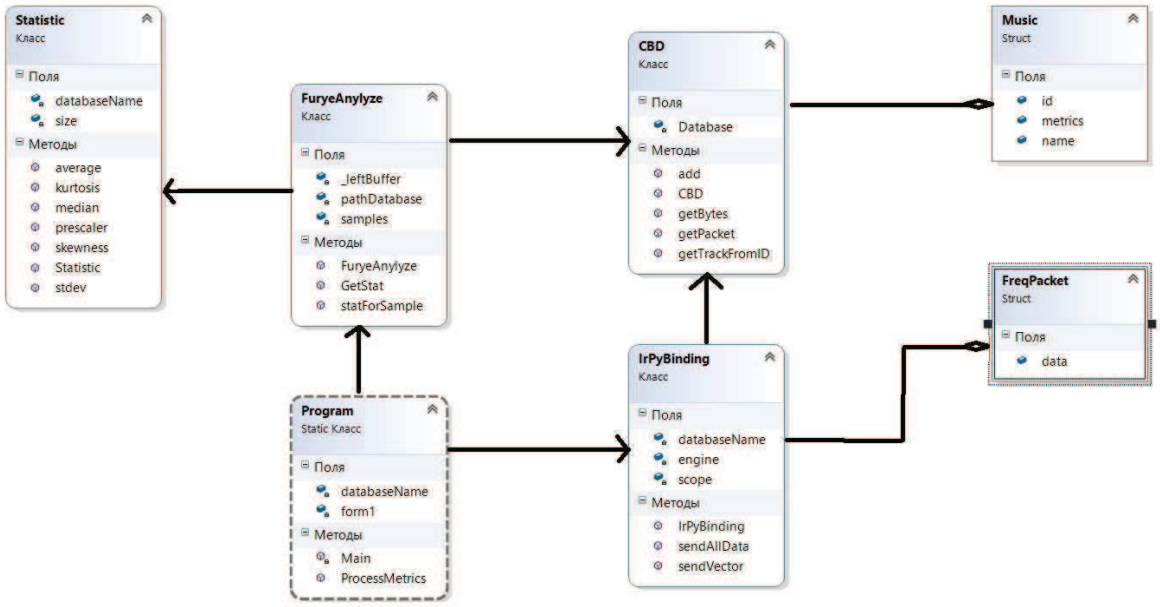


Рисунок 29. Объектная модель анализатора музыкальных предпочтений.

Проведя анализ всего вышесказанного, создается объектная модель программного продукта, представленная на рисунке 29.

Структурами хранящими музыкальные треки являются FreqPacket (необходим для записи вектора данных в БД) и Music, в которой будут храниться указатели для поиска песни в БД, ее метаданные и обучающий вектор для нейросети.

Для хранения объектов предметной области использоваться тип данных CBD, он будет выполнять работу с данными о треках пользователя в СУБД, в качестве которой выбрана интегрированная sqllite.

Для получения статистик по звуковому ряду используется класс Statistic, который представляет собой статический класс для математических вычислений.

Класс IryPYBinding включает в себя набор свойств для работы с аналитическим модулем программы, непосредственно нейросетью, написанной на языке Python.

Основным классом в иерархии модуля получения данных является класс FuryeAnalyze. В нем отображены отношения между остальными обьектами модуля.

Получив объектную модель системы, можно перейти непосредственно к реализации программного продукта. Объектная модель представляется на языке программирования C#.

Глава 4. РЕАЛИЗАЦИЯ

4.1 Реализация блока сбора статистики

На рисунке 30 приведена блок-схема алгоритма перевода аудиофайла в вектор значений.

Рисунок 30. Блок-схема алгоритма перевода аудиофайла в вектор значений

Для извлечения амплитуд сигнала во времени из файла mp3 используется сторонняя библиотека **NAudio**.

NAudio — OpenSource библиотека для работы со звуком на платформе .NET, является «оберткой» для системных API, упрощающей разработку.

В среде разработки Visual Studio можно использовать пакетный менеджер NuGet для подключения библиотеки к проекту. В окне пакетного менеджера следует ввести команду:

Install-Package NAudio

Работа с библиотекой выглядит следующим образом:

```
using (var reader = new Mp3FileReader(fileName))
        {
            var pcmLength = (int)reader.Length;
            _leftBuffer = new byte[pcmLength / 2];
            var buffer = new byte[pcmLength];
            var bytesRead = reader.Read(buffer, 0, pcmLength);

            int index = 0;
            for (int i = 0; i < bytesRead; i += 4)
            {
                _leftBuffer[index] = buffer[i];
                index++;
                _leftBuffer[index] = buffer[i + 1];
                index++;
            }

        }
```

В массиве байт _leftBuffer хранится 1 канал звуковой дорожки. Для анализа нам достаточно одного канала.

Поскольку амплитуда в файле формата mp3 кодируется 2 байтами, следует провести еще одно преобразование.

```
var waveBuffer = new WaveBuffer(_leftBuffer);

var amplitude = waveBuffer.ShortBuffer[AmplitudeIndex];
```

Теперь в переменной amplitude хранится значение амплитуды с индексом AmplitudeIndex.

Имея массив амплитуд сигнала во времени можно разбить его на части по 4096 значений и к каждой части последовательно применить оконную функцию и метод преобразования Фурье.

Для преобразования Фурье используется сторонняя библиотека **Meta.Numerics**. Эта библиотека для платформы .NET поддерживает математическую статистику, матричные преобразования и специальные математические функции, коей является преобразование Фурье. Добавить эту библиотеку в проект можно с помощью пакетного менеджера NuGet следующей командой:

Install-Package Meta.Numerics

Преобразование выглядит следующим образом:

```
FourierTransformer ft = new FourierTransformer(4096);

Complex[] data_interval = new Complex[4096];

Complex comp = new Complex(waveBuffer.ShortBuffer[i * 2048 + k], 0);

data_interval[k] = comp;

Complex[] xt = ft.Transform(data_interval);
```

В итоге получаем массив комплексных чисел xt, из которого можно извлечь магнитуду для частоты из дискретного набора следующим образом:

```
magnitude = Math.Pow(MoreMath.Sqr(c.Re) + MoreMath.Sqr(c.Im), 0.5)/2048;
```

Полученные значения пишутся в матрицу, в которой 1 индекс - это разрешение по частоте, а второе по времени (номер фрейма).

Следующим этапом является расчет метрик для каждой последовательности магнитуд частоты из дискретного набора. Для расчета статистик используется собственный класс Statistic. Вот реализации его методов для их расчета:

```
public double median(ref double[][] frame, int index) // медиана
 {
     Array.Sort(frame[index]);
     return (frame[index][size/2] + frame[index][size/2 - 1])/2;
}

 public double average(ref double[][] frame, int index) // среднее
```

```
{
    return frame[index].Average();
}

 public double stdev(ref double[][] frame, int index, double average) // дисперсия
{
     double Sum = 0;
     double n = frame[index].Length - 1;

     foreach (double x in frame[index])
     {
         double k = MoreMath.Sqr(x - average)/n;
         Sum += k;
      }


      return Math.Pow(Sum,0.5);
 }


public double skewness(ref double[][] frame, int index, double stdev, double
average) // скос (ассиметрия)
{
     double Sum = 0;
     double n = frame[index].Length;

        foreach ( double x in frame[index])
       {
           Sum += Math.Pow(((x - average)/stdev), 3);
        }


        return Sum/n;
 }

public double kurtosis(ref double[][] frame, int index , double stdev, double
average) // крутизна (эксцесс)
{
       double Sum = 0;
       double n = frame[index].Length;

       foreach (double x in frame[index])
      {
           Sum += Math.Pow(((x - average)/stdev), 4);
```

```
        }

        return Sum/n;
    }
```

Для хранения векторов используется БД **SQLite**. SQLite — это встраиваемая кроссплатформенная БД, которая поддерживает достаточно полный набор команд SQL и доступна в исходных кодах (на языке C). Исходные коды SQLite находятся в public domain, то есть не имеют никаких ограничений на использование. Добавить БД в проект можно с помощью пакетного менеджера NuGet командой:

Install-Package System.Data.SQLite

За взаимодействие с БД отвечает собственный класс CBD. Для добавления вектора в тестовую базу используется метода add(). Вот его реализация:

```
public void add(string trackName, ref FreqPacket arrayF)
    {
        byte[] array = getBytes(ref arrayF);
        int mark = 0;


        DirectoryInfo d = new
        DirectoryInfo(Path.GetDirectoryName(trackName));
        string markString = d.Name;

        if (!Int32.TryParse(markString, out mark))
            mark = 2;


        try
        {
            using (SQLiteConnection connection = new
        SQLiteConnection(string.Format("Data Source={0};", Database)))
            {
                connection.Open();

                using (SQLiteTransaction transaction =
        connection.BeginTransaction())
                {
                    using (SQLiteCommand command = new SQLiteCommand("CREATE TABLE
IF NOT EXISTS 'MUSIC' (id INTEGER PRIMARY KEY AUTOINCREMENT, track TEXT,
param BLOB, mark INTEGER);", connection))
                    {
```

```
                    command.ExecuteNonQuery();

                    command.CommandText = "INSERT INTO 'MUSIC' ('track','param','mark')
VALUES (@trackName,@array,@mark)";
                    command.Parameters.Add("@array", DbType.Binary, array.Length).Value =
array;
                    command.Parameters.Add("@trackName", DbType.String, trackName.Length
* sizeof(char)).Value = trackName;
                    command.Parameters.Add("@mark", DbType.Int32, sizeof(int)).Value = mark;
                    command.ExecuteNonQuery();
                }

                transaction.Commit();
            }

            connection.Close();
        }

    }
    catch (SQLiteException ex)
    {
      if (ex.ResultCode.Equals(SQLiteErrorCode.Busy))
          Console.WriteLine("Database is locked by another process!");
    }
}
```

Структура FreqPacket, в которой хранится вектор трека преобразуется с помощью статического метода getBytes(ref arrayF) класса CBD в бинарный формат, чтобы в дальнейшем записать его в БД в поле BLOB. Затем в БД создается таблица со следующими полями: id INTEGER PRIMARY KEY AUTOINCREMENT, track TEXT, param BLOB, mark INTEGER. Где track - это название трека, mark - оценка, а param - вектор в двоичном представлении.

Запись в БД ведется с использованием транзакций:

```
SQLiteTransaction transaction = connection.BeginTransaction();

/*****************
//Блок транзакции
*****************/
transaction.Commit();
```

Для извлечения информации из БД по индексу существует метод getTrackFromID(). К тому же для получения информации о количестве треков в тестовой таблице существует метод getCountTrack().

Использование транзакции обусловлено тем, что одновременно может быть несколько подключений к БД из разных потоков.

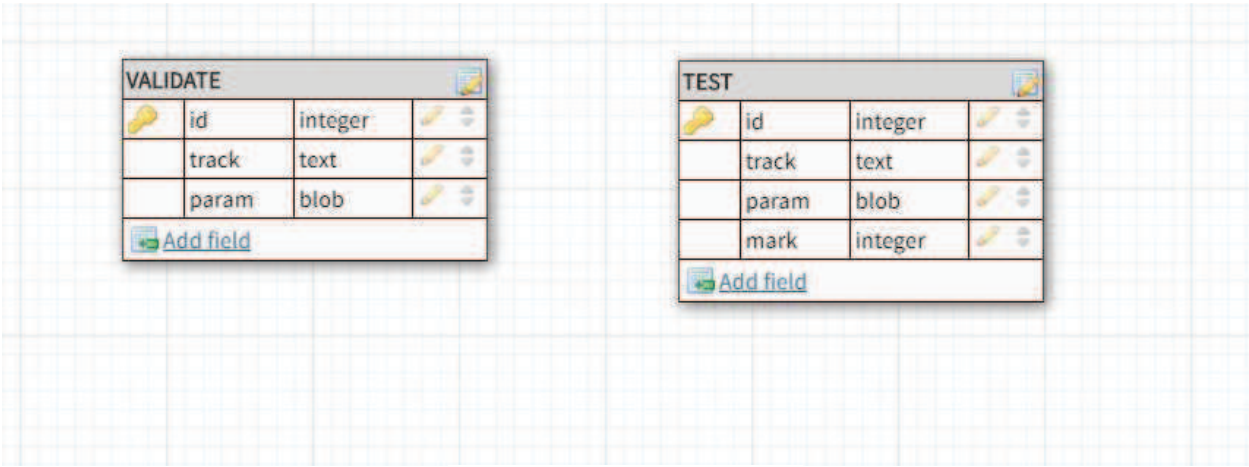


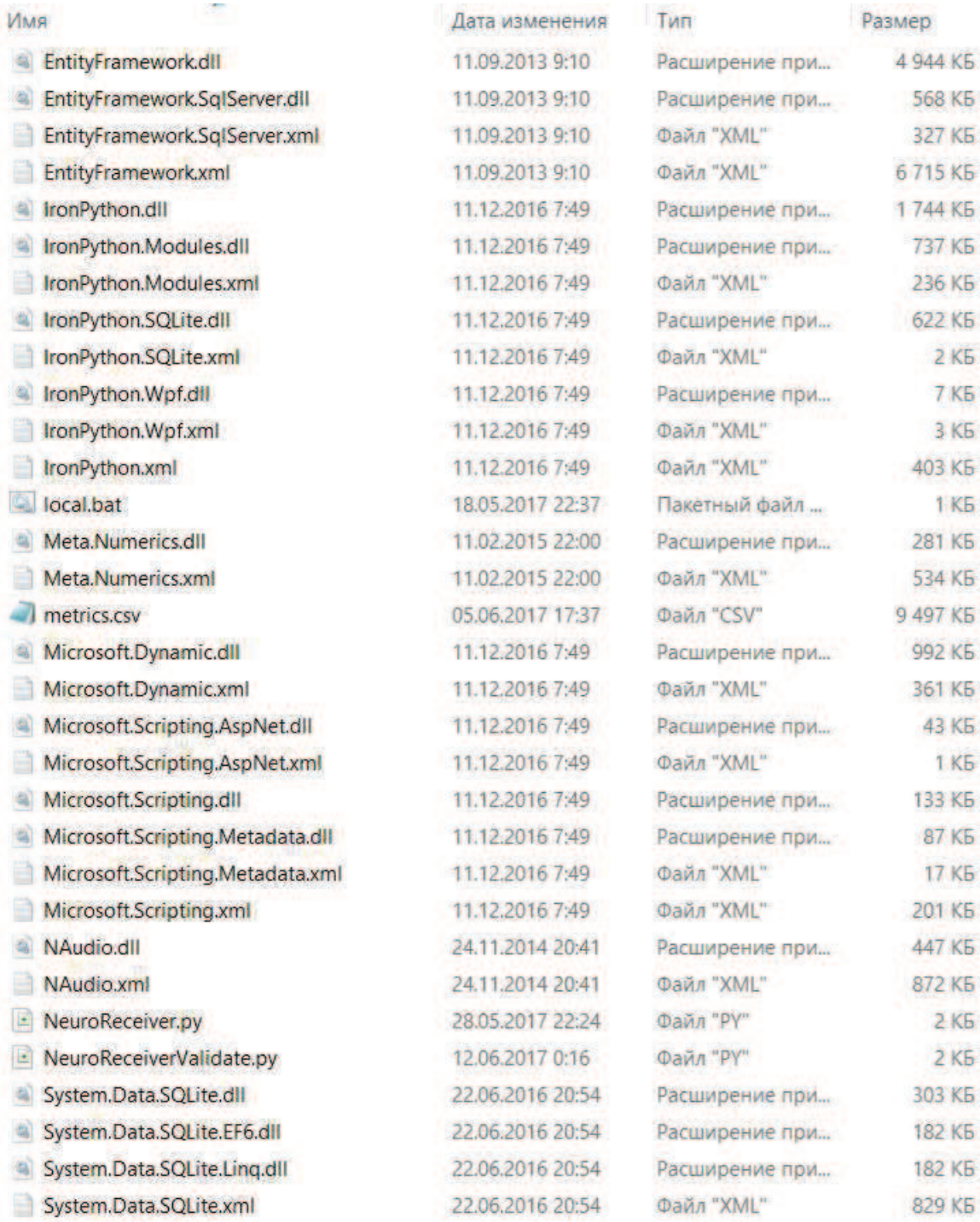
Рисунок 31. Схема БД

Для ускорения процесса сбора статистики код был распараллелен с использованием пула потоков (класс ThreadPool). Для того чтобы не создавалось слишком много потоков используется следующий механизм блокировки:

```
foreach (string s in files)
            {
               while (locker <= 0)
               {

                  Application.DoEvents();
               }
               lock (Lock)
               {
                  locker--;
               }
               try
               {

                  ThreadPool.QueueUserWorkItem(new WaitCallback(delegate
(object state)
                  {
                     object[] array = state as object[];
                     FuryeAnylyze FA = new FuryeAnylyze(array[0].ToString());
                     FA.GetStat(array[1].ToString());

                     GC.Collect(GC.MaxGeneration, GCCollectionMode.Forced);
```

```
                    form1.getListView().BeginInvoke(new InvokeDelegate(delegate
()
                    {
                        form1.getListView().Items.Add((new
ListViewItem(form1.getListView().Items.Count.ToString() + ". " +
DateTime.Now.ToString() + " - " + array[1].ToString())));
                    }));


                    lock (Lock)
                    {
                        locker++;
                    }


                }), new object[] { databaseName, s });
            }

            catch (Exception exception)
            {
                Console.WriteLine(exception.Message);

            }


        }
```

Количество потоков ограничено блокировками до 4 штук. При достижения этой отметки новый поток не добавляется в пул. Работа из потоков с формами ведется с помощью специальных методов BeginInvoke().



Имя	Дата изменения	Тип	Размер
EntityFramework.dll	11.09.2013 9:10	Расширение при...	4 944 КБ
EntityFramework.SqlServer.dll	11.09.2013 9:10	Расширение при...	568 КБ
EntityFramework.SqlServer.xml	11.09.2013 9:10	Файл "XML"	327 КБ
EntityFramework.xml	11.09.2013 9:10	Файл "XML"	6 715 КБ
IronPython.dll	11.12.2016 7:49	Расширение при...	1 744 КБ
IronPython.Modules.dll	11.12.2016 7:49	Расширение при...	737 КБ
IronPython.Modules.xml	11.12.2016 7:49	Файл "XML"	236 КБ
IronPython.SQLite.dll	11.12.2016 7:49	Расширение при...	622 КБ
IronPython.SQLite.xml	11.12.2016 7:49	Файл "XML"	2 КБ
IronPython.Wpf.dll	11.12.2016 7:49	Расширение при...	7 КБ
IronPython.Wpf.xml	11.12.2016 7:49	Файл "XML"	3 КБ
IronPython.xml	11.12.2016 7:49	Файл "XML"	403 КБ
local.bat	18.05.2017 22:37	Пакетный файл ...	1 КБ
Meta.Numerics.dll	11.02.2015 22:00	Расширение при...	281 КБ
Meta.Numerics.xml	11.02.2015 22:00	Файл "XML"	534 КБ
metrics.csv	05.06.2017 17:37	Файл "CSV"	9 497 КБ
Microsoft.Dynamic.dll	11.12.2016 7:49	Расширение при...	992 КБ
Microsoft.Dynamic.xml	11.12.2016 7:49	Файл "XML"	361 КБ
Microsoft.Scripting.AspNet.dll	11.12.2016 7:49	Расширение при...	43 КБ
Microsoft.Scripting.AspNet.xml	11.12.2016 7:49	Файл "XML"	1 КБ
Microsoft.Scripting.dll	11.12.2016 7:49	Расширение при...	133 КБ
Microsoft.Scripting.Metadata.dll	11.12.2016 7:49	Расширение при...	87 КБ
Microsoft.Scripting.Metadata.xml	11.12.2016 7:49	Файл "XML"	17 КБ
Microsoft.Scripting.xml	11.12.2016 7:49	Файл "XML"	201 КБ
NAudio.dll	24.11.2014 20:41	Расширение при...	447 КБ
NAudio.xml	24.11.2014 20:41	Файл "XML"	872 КБ
NeuroReceiver.py	28.05.2017 22:24	Файл "PY"	2 КБ
NeuroReceiverValidate.py	12.06.2017 0:16	Файл "PY"	2 КБ
System.Data.SQLite.dll	22.06.2016 20:54	Расширение при...	303 КБ
System.Data.SQLite.EF6.dll	22.06.2016 20:54	Расширение при...	182 КБ
System.Data.SQLite.Linq.dll	22.06.2016 20:54	Расширение при...	182 КБ
System.Data.SQLite.xml	22.06.2016 20:54	Файл "XML"	829 КБ

Рисунок 32. Окружение программы

4.2 Реализация аналитического блока

Анализатор реализован на языке Python с использованием библиотеки Pybrain. PyBrian представляет собой модульную библиотеку предназначенную для реализации различных алгоритмов машинного

обучения на языке Python. Основной его целью является предоставление исследователю гибких, простых в использовании, но в то же время мощных инструментов для реализации задач из области машинного обучения, тестирования и сравнения эффективности различных алгоритмов.

Библиотека построена по модульному принципу, что позволяет использовать её как студентам для обучения основам, так и исследователям, нуждающимся в реализации более сложных алгоритмов. Общая структура процедуры её использования приведена на рисунке 33. Сама библиотека является продуктом с открытым исходным кодом и бесплатна для использования в любом проекте.

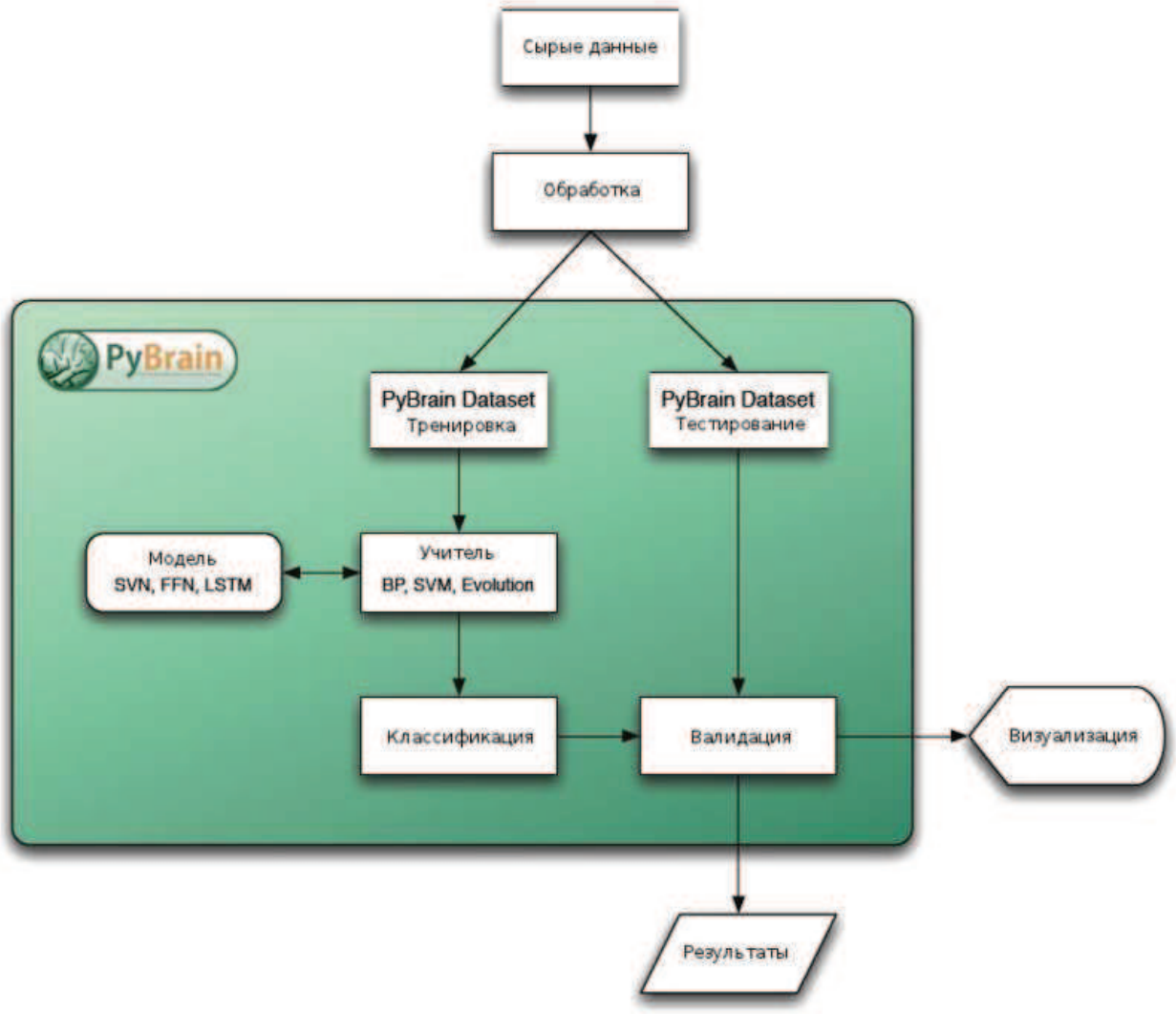


Рисунок 33. Структура модульной библиотеки Pybrain

Рисунок 34. Алгоритм работы анализатора

Данные анализатор берет из файла в формате csv. В строке содержатся 1000 входных параметров и класс, к которому отнесен пользователь. Этот файл извлекается в векторы X и Y, которые представляют собой входные и выходные данные.

Реализация метода извлечения данных:

```
def loadDataFromCSV(X,Y):


    tf = open('metrics.csv','r')

    for line in tf.readlines():
        data = [float(x) for x in line.strip().split(',') if x != '']
        indata =  tuple(data[:1000])
        outdata = tuple(data[1000:])
        X.append(indata)
        Y.append(outdata)

    return
```

Следующим этапом является нормализация данных для приведения их к интервалу значений [0,1]. Нормализации подвергается вектор входных данных сети X.

Реализация метода нормализации данных:

```
def Normilize(X,Xmax,Xmin):


    for i in range(len(X)):
        X[i]=(X[i] - Xmin)/(Xmax - Xmin)


    return
```

Данные необходимо отформатировать прежде чем создавать обучающее множество. Реализация форматирования с помощью библиотеки numpy:

```
import numpy as np


X = np.array(X).reshape(dataSize,1000)
Y = np.array(Y).reshape(dataSize,1)
```

Теперь можно создавать обучающее множество. Для этого нам понадобится объект pybrain.datasets.ClassificationDataSet, используемый в задачах классификации. Сделаем вспомогательную функцию, которая наполняет датасет, получив на вход матрицы входных и выходных значений:

```
def buildDataSet(X,Y):
    data = ClassificationDataSet(1000, 1, nb_classes=4, class_labels=['D', 'C', 'B', 'A'])

    for x,y in zip(X,Y):
        data.addSample(x, y)

    return data
```

Разделим наши данные на обучающую и тестовую выборки с помощью sklearn и сделаем из них датасеты. Разбиение происходит случайным образом, но с сохранением баланса в выборках.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y)

train = buildDataSet(X_train, y_train)

test = buildDataSet(X_test, y_test)
```

Нам следует преобразовать выход таким образом, чтобы каждому классу соответствовал свой столбец (нейросети так лучше обучаются). То есть мы хотим превратить [0] в [1, 0] и [1] в [0, 1]. Для этого можно использовать такой метод:

```
train._convertToOneOfMany()
test._convertToOneOfMany()
```

Теперь создадим нейросеть. Библиотека *pybrain* позволяет создавать достаточно сложные нейросети с различным числом слоёв и хитро устроенными связями между ними. Но мы хотим использовать простейшую

сеть с одним скрытым слоем и поэтому воспользуемся готовой функцией pybrain.tools.shortcuts.buildNetwork. Впрочем, нам понадобится ей указать, какие активирующие функции мы хотим использовать: а именно, для среднего слоя нейронов мы хотим использовать гиперболический тангенс, а для вычисления итоговых вероятностей — softmax. Чтобы это сделать, нам нужно загрузить соответствующие объекты.

```
from pybrain.tools.shortcuts import buildNetwork

from pybrain.structure import TanhLayer, SoftmaxLayer


input_neurons = 1000

output_neurons = 500

hidden_neurons = 4


net = buildNetwork(input_neurons, hidden_neurons, output_neurons,

                   hiddenclass=TanhLayer, outclass=SoftmaxLayer)
```

Теперь потребуется создать trainer — объект, который и будет обучаться.

```
from pybrain.supervised.trainers import BackpropTrainer
trainer = BackpropTrainer(net, train, weightdecay=0.0001)
# weightdecay — это параметр регуляризации
```

Параметр регуляризации укажем равным 0.0001, что является приемлемым для количества обучающих данных в данной задаче.

Для начала процесса обучения существует функция trainEpochs(). В качестве параметра используется количество эпох для обучения. Экспериментально получено, что данная сеть обучается за 500 эпох.

```
trainer.trainEpochs(500)
```

После того, как сеть обучиться, следует сериализовать ее, используя библиотеку pickle.

```
import pickle

fileObject = open('net', 'w')

pickle.dump(net, fileObject)

fileObject.close()
```

4.3 Реализация связующего блока

Модуль сбора статистики и аналитический модуль на писаны на разных языках программирования (C# и Python). По этой причине следует реализовать механизм обмена данными. В качестве такого механизма выбраны сокеты на языке Python. При этом, в платформе .NET существует, поддерживаемый ей язык - IronPython, который способен встраиваться в код на C#. Используя классы "обертки" ScriptEngine и ScriptScope, можно использовать скрипты на языке IronPython прямо в коде на языке C#.

Реализация использования ScriptEngine и ScriptScope:

```
using IronPython.Hosting;
using Microsoft.Scripting.Hosting;

ScriptEngine engine;
ScriptScope scope;


engine = Python.CreateEngine();
var paths = engine.GetSearchPaths();
paths.Add(@"C:\AudioAnalyzer\Lib");
engine.SetSearchPaths(paths);
scope = engine.CreateScope();


scope.SetVariable("doubleArray", data);
scope.SetVariable("mark", (double)mark);
engine.ExecuteFile(@"IronPythonBinding.py", scope);
```

В файле IronPythonBinding.py содержится реализация обмена данными на основе сокетов с модулем анализатора.

Сокет — это программный интерфейс для обеспечения информационного обмена между процессами. Существуют клиентские и серверные сокеты. Серверный сокет прослушивает определенный порт, а

клиентский подключается к серверу. После того, как было установлено соединение начинается обмен данными.

После успешной установки соединения сервер и клиент начинают обмениваться информацией. Сервером выступает аналитический модуль. Клиентская часть посылает размер данных и затем сами данные. Сервер в свою очередь принимает вектора с оценками (посылку из 1001 цифры) и отсылает подтверждение. В качестве порта выбран порт номер 9090.

4.4 Реализация системы рекомендаций

Для выставления оценок пользователем выбирается директория с новыми треками. Для этих треков ведется преобразование каждого из них в вектор и отправка их на проверку (валидацию).

В качестве валидатора выступает скрипт на языке Python. Проверка осуществляется на базе ранее обученной сети. Как и в случае обучения тестовые данные подвергаются нормировке и форматированию. Затем формируется тестовое множество, которое активируется функцией activateOnDataset() на обученной сети.

```
fileObject = open('net','r')
net = pickle.load(fileObject)

trainer = BackpropTrainer(net)

test = buildDataSet(X, Y)
result_on_test = trainer.module.activateOnDataset(test)
```

В итоге, в result_on_test находятся вероятности попадания трека в определенную группу предпочтений.

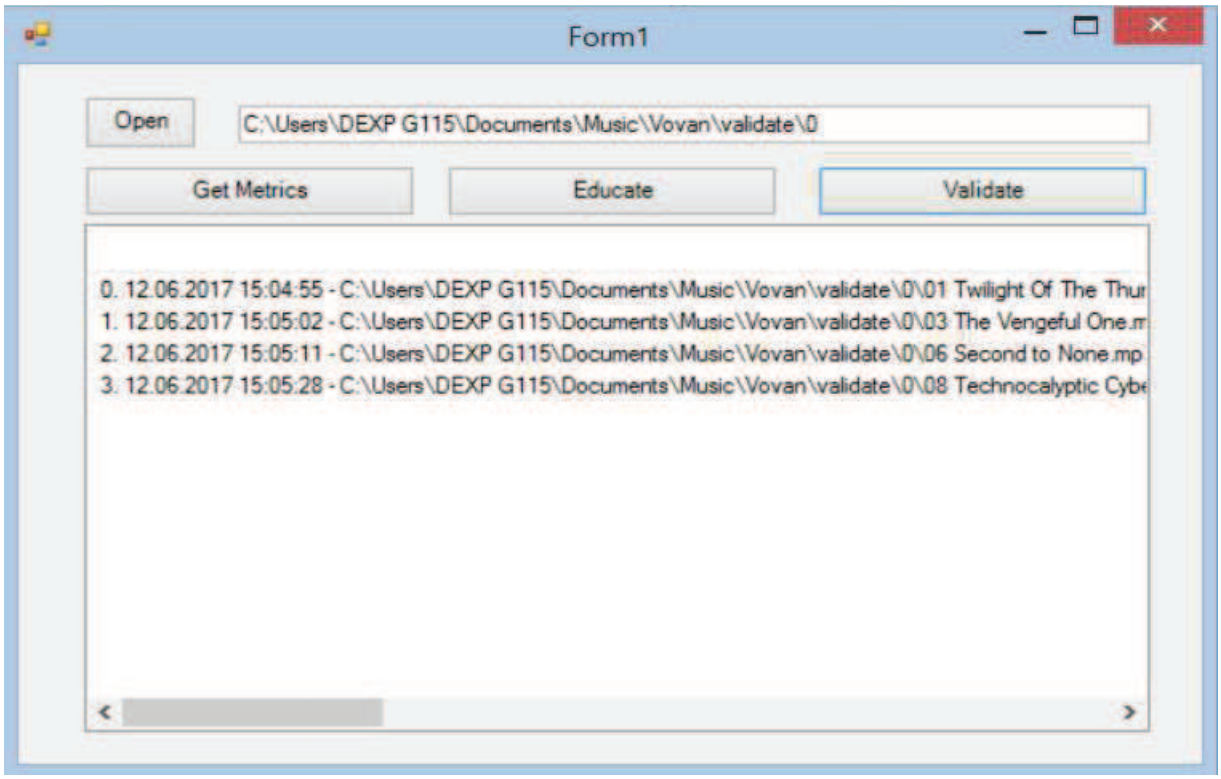


Рисунок 35. Программный интерфейс

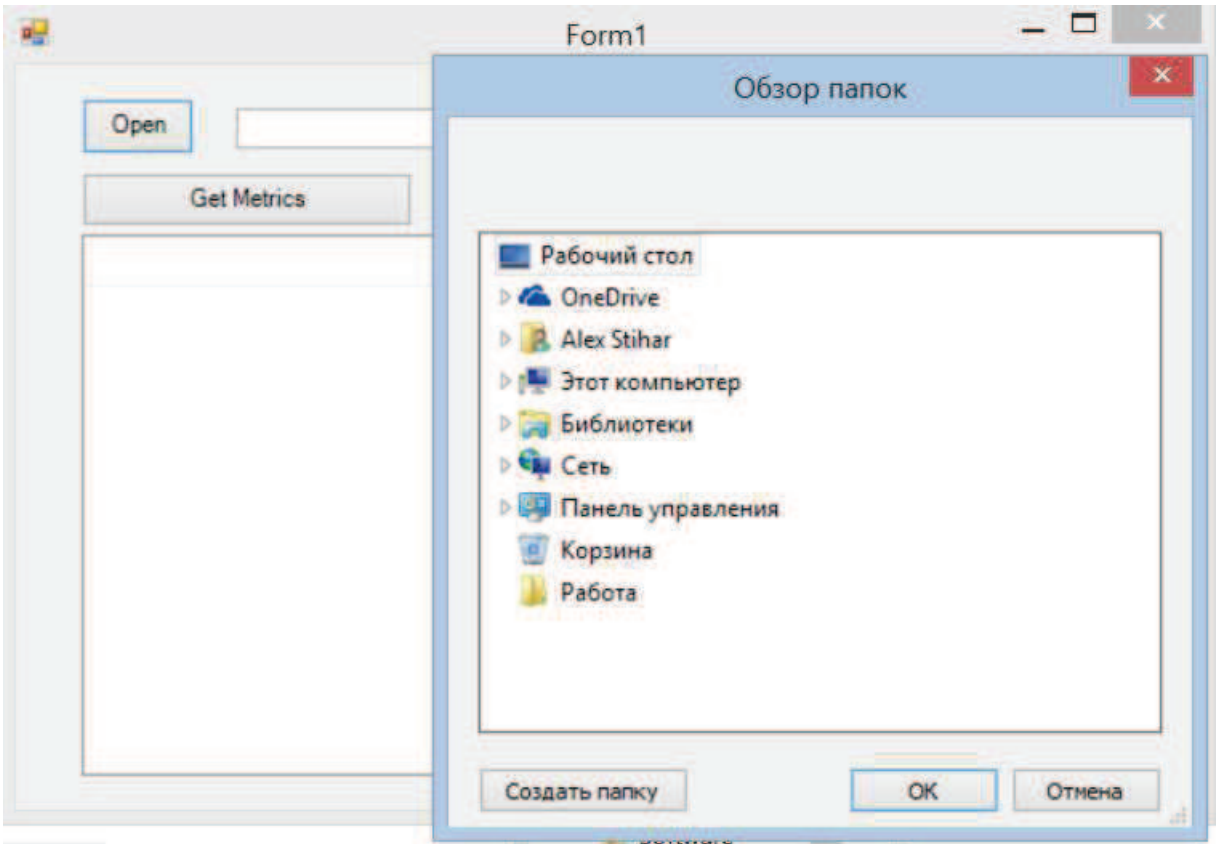


Рисунок 36. Выбор директории

Глава 5. ОЦЕНКА РЕЗУЛЬТАТОВ РАБОТЫ

Оценка эффективности работы программы оценивается на базе тестовой и валидационной выборок.

При обучении нейронной сети часть векторов (132 из 528) ушло в контрольную выборку и в процессе обучения не учувствовали. При этом, после каждой эпохи обучения сеть активировалась на данной тестовой выборки и просчитывался процент ошибок на ней. Данная методика часто применяется при обучении нейронных сетей и позволяет оценить ее качество на реальных данных.

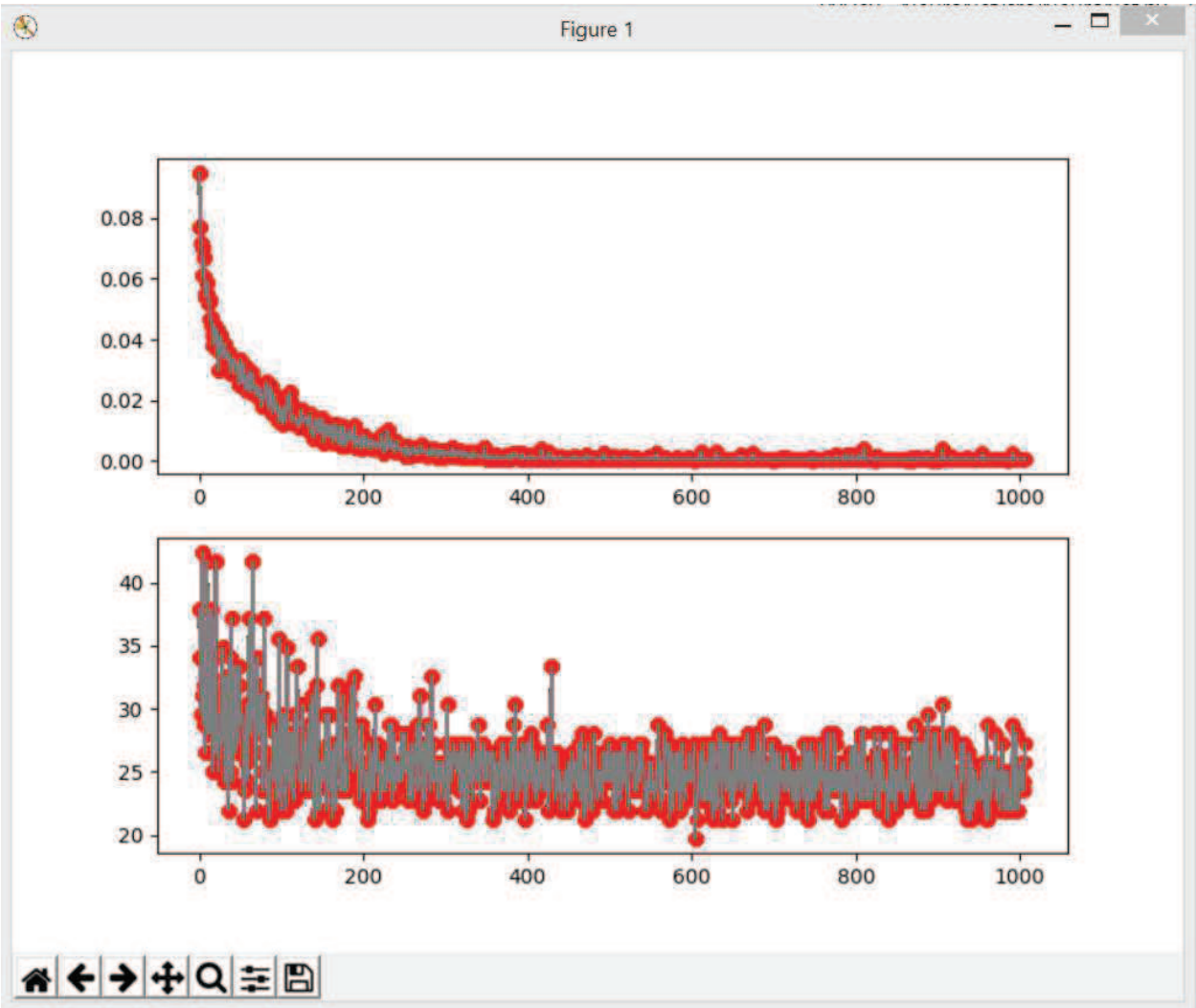


Рисунок 37. Контроль обучения

На основании рисунка 37 можно увидеть на верхнем графике, что в процессе обучения сеть сходится, что говорит о постепенном уменьшении ошибки на обучающей выборке и успешном течении процесса обучения. среднеквадратичная ошибка на выходах сети по результатам 1000 итераций составила 0.0005, что для данной задачи является приемлемым результатом.

Формула для расчета среднеквадратичной ошибки:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}$$

Особый интерес представляет нижний график. На нем видно, что в результате процесса обучения процент ошибок на тестовой выборки вошел в коридор 20-30 %, что говорит о том, что нейронной сети удалось проявить свои обобщающие способности и выделить в своем внутреннем представлении классы.

Проанализируем ошибки. По таблице ниже видно, что в связи с тем, что 3 пользователь не может быть оценен человеком, так как музыка для этого класса была собрана случайным образом, анализу подлежат 40 ошибок. но так как ошибка учтена дважды, в качестве входящей и исходящей, то анализу пользователей подвергнутся 20 треков.

Пользователь	Входящие ошибки	Исходящие ошибки	Суммарно Ошибок	Не подлежащие оценки	Ошибки подлежащие оценке
4	17	3	20	3	17
3	17	7	24	17	7
2	2	20	22	7	15
1	1	7	8	7	1

По результатов оценивания людьми треков, которые система ошибочно отнесла к другому классу можно сделать следующую таблицу:

Пользователь	Прослушано треков	Положительная оценка
4	17	12
2	2	2
1	1	0

На основании опроса пользователей оказалось, что грубыми оценками является 30% из общего числа ошибок. Учитывая процент ошибок равный 28% на тестовом наборе, можно говорить о 8.4 % процентах грубых ошибок.

ЗАКЛЮЧЕНИЕ

В данной работе был разработан продукт, предназначенный для классификации музыки согласно музыкальным предпочтениям слушателей. Простой интерфейс программы не вызовет трудности для начинающего пользователя. Данная программа не использует в своем анализе ничего, кроме формы звуковой волны музыкального трека.

В ходе выполнения работы решены следующие задачи:

1. Проведен детальный анализ методов, решений и характеристик в области аудиоаналитики;
2. Разработана система аудиоанализа музыкальных коллекций исключительно на основе звуковой волны, способную успешно решать задачу прогнозирования оценки пользователя на аудиотрек;
3. Проведен анализ результатов работы системы.

Результат работы – программный продукт, который позволяет прогнозировать оценку пользователя перед прослушиванием аудиокомпозиции с ошибкой в 8.4 % .

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Механизм работы Genius в iTunes [Электронный ресурс]. – Режим доступа: http://www.macdigger.ru/macall/sekrety-raboty-genius-v-itunes.html. – Заглавие с экрана. – (Дата обращения: 16.05.2017).
2. Рекомендательная технология Диско [Электронный ресурс]. – https://yandex.ru/blog/company/9691 . – Заглавие с экрана. – (Дата обращения: 16.05.2017).
3. Машинный слух. SoundNet [Электронный ресурс]. – Режим доступа: https://geektimes.ru/post/283332/. – Заглавие с экрана. – (Дата обращения: 16.05.2017).
4. Как мозг выбирает музыку [Электронный ресурс]. – Режим доступа: https://www.nkj.ru/news/22012/. – Заглавие с экрана. – (Дата обращения: 16.05.2017).
5. Цифровой слух. Как компания Яндекс подбирает музыку под ваше настроение [Электронный ресурс]. – Режим доступа: http://www.forbes.ru/forbeslife/344029-cifrovoy-sluh-kak-yandeks-podbiraet-muzyku-pod-vashe-nastroenie. – Заглавие с экрана. – (Дата обращения: 16.05.2017).
6. Анализ алгоритмов аудиоаналитики [Электронный ресурс]. – Режим доступа: https://habrahabr.ru/company/synesis/blog/250935/. – Заглавие с экрана. – (Дата обращения: 16.05.2017).
7. Tom Schaul, Justin Bayer, Daan Wierstra, Sun Yi, Martin Felder, Frank Sehnke, Thomas Rückstieß, Jürgen Schmidhuber. PyBrain. To appear in: Journal of Machine Learning Research, 2010.
8. Нейронные сети от теории к практике [Электронный ресурс]. – Режим доступа: https://www.mql5.com/ru/articles/497. – Заглавие с экрана. – Англ. – (Дата обращения 16.05.2017).
9. Глубокое обучение нейронной сети. [Электронный ресурс]. – Режим доступа: https://habrahabr.ru/company/wunderfund/blog/315476/. – Заглавие с экрана. – (Дата обращения: 23.05.2017).
10. Проектирование многослойного персептрона. [Электронный ресурс]. – Режим доступа: http://matlab.exponenta.ru/neuralnetwork/book4/3_2.php. – Заглавие с экрана. – (Дата обращения: 23.05.2017).
11. Простой классификатор на Pybrain. [Электронный ресурс]. – Режим доступа: https://habrahabr.ru/post/171937/. – Заглавие с экрана. – (Дата обращения: 17.05.2017).

12. Обзор библиотеки Pybrain. [Электронный ресурс]. – Режим доступа: https://habrahabr.ru/post/148407/. – Заглавие с экрана. – (Дата обращения: 17.05.2017).
13. Курс по машинному обучению [Электронный ресурс]. – Режим доступа: https://github.com/ischurov/hse_nes_ml/blob/master/lesson4_neural.ipynb. – Заглавие с экрана. – (Дата обращения 16.05.2017).
14. Гмурман В. Е. Руководство к решению задач по теории вероятностей и математической статистике. - М., Высш.шк., 2004.- 404 с
15. Хейлсберг А. Язык программирования C# / Хейлсберг А., Торгерсен М.4-е изд. – СПБ.: Питер, 2012. – 708 с.
16. Прохоренок Н. Python 3 / Н. Прохоренок, В. Дронов. – СПб.: БХВ-Петербург, 2016. – 464 с.

ПРИЛОЖЕНИЕ

исходный код исполняемого модуля (дебаг версия) на языках C# и Python

```
//*********************************

//Program.cs

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Threading;
using System.Windows.Forms;
using NAudio;
using System.Runtime.InteropServices;
using System.Diagnostics;
using System.Data.SQLite;
using System.Data.Common;
using System.IO;

namespace AudioAnalyzer
{
    [StructLayout(LayoutKind.Sequential)]
    public struct FreqPacket
    {
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 10240)]
        public double[] data;
    }



    public struct Music
    {
        public long id;
        public string name;
        public double[] metrics; // 200 * 5 metrics
        public long mark;
    }



    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        ///


        static Form1 form1 = new Form1();
        static string databaseName = @"C:\Database\cyber.db";
        static string databaseValidate = @"C:\Database\validate.db";
        public delegate void InvokeDelegate();
        static Object Lock = new Object();

        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
```

```
            string dirName = @"C:\Users\DEXP G115\Dropbox\music";


            Application.Run(form1);


        }


        static public void Validate()
        {


            if (Directory.Exists(form1.GetDir()))
            {
                string[] files = Directory.GetFiles(form1.GetDir());


                foreach (string s in files)
                {


                    try
                    {

                            FuryeAnylyze FA = new FuryeAnylyze(databaseValidate);
                            FA.GetStat(s);


                            form1.getListView().BeginInvoke(new
InvokeDelegate(delegate ()
                            {
                                form1.getListView().Items.Add((new
ListViewItem(form1.getListView().Items.Count.ToString() + ". " +
DateTime.Now.ToString() + " - " + s)));
                            }));

                            Application.DoEvents();


                    }

                    catch (Exception exception)
                    {
                        Console.WriteLine(exception.Message);

                    }

                }

                IrPyBinding irpb = new IrPyBinding(databaseValidate);
                irpb.sendValidate();
            }
        }
        static public void Educate()
        {
            ThreadPool.QueueUserWorkItem(new WaitCallback(delegate (object state) {
```

```
                IrPyBinding irpb = new IrPyBinding(databaseName);
                irpb.sendAllData();

            }));

        }
        static public void ProcessMetrics()
        {
            ThreadPool.SetMaxThreads(2, 2);
            int locker = 4;

            if (Directory.Exists(form1.GetDir()))
            {
                string[] files = Directory.GetFiles(form1.GetDir());


                foreach (string s in files)
                {
                    while (locker <= 0)
                    {

                        Application.DoEvents();
                    }

                    lock (Lock)
                    {
                        locker--;
                    }
                    try
                    {

                        ThreadPool.QueueUserWorkItem(new WaitCallback(delegate
(object state)
                        {
                            object[] array = state as object[];
                            FuryeAnylyze FA = new FuryeAnylyze(array[0].ToString());
                            FA.GetStat(array[1].ToString());

                            GC.Collect(GC.MaxGeneration, GCCollectionMode.Forced);


                            form1.getListView().BeginInvoke(new
InvokeDelegate(delegate ()
                           {
                               form1.getListView().Items.Add((new
ListViewItem(form1.getListView().Items.Count.ToString() + ". " +
DateTime.Now.ToString() + " - " + array[1].ToString())));
                           }));


                            lock (Lock)
                            {
                                locker++;
                            }

                        }), new object[] { databaseName, s });
                    }

                    catch (Exception exception)
                    {
                        Console.WriteLine(exception.Message);
```

```
                    }


                }



            }
        }
    }
}


//*******************************************************
//FuryaAnylyze.cs

using System;
using System.Collections.Generic;
using System.Linq;
using System.Collections;
using System.Text;
using System.Threading.Tasks;
using NAudio;

using System.Runtime.InteropServices;
using System.Data.SQLite;
using System.Data.Common;
using System.IO;
using NAudio.Wave;
using Meta.Numerics.SignalProcessing;
using Meta.Numerics;
using System.Globalization;

namespace AudioAnalyzer
{
    class FuryeAnylyze
    {
        private string pathDatabase;
        private byte[] _leftBuffer;
        short[] samples;

        public FuryeAnylyze(string databaseName)
        {
            pathDatabase = databaseName;

        }

        public static bool statForSample(ref double[][] data, string databaseName,
string fileName)
        {

            Statistic stat = new Statistic(data[0].Length, databaseName);
            FreqPacket FP = new FreqPacket();
            CBD cbd = new CBD(databaseName);
            FP.data = new double[10240];
            int k = 0;

                for (int i = 0; i < 2048; i++)
```

```
                {

                    double avg = stat.average(ref data, i);// для частоты k-той во
времени
                    double stdev = stat.stdev(ref data, i, avg);
                    double med = stat.median(ref data, i);
                    double skewness = stat.skewness(ref data, i, stdev, avg);
                    double kurtosis = stat.kurtosis(ref data, i, stdev, avg);

                    FP.data[k] = avg;
                    FP.data[k + 1] = stdev;
                    FP.data[k + 2] = med;
                    FP.data[k + 3] = skewness;
                    FP.data[k + 4] = kurtosis;

                    k += 5;

                }

                cbd.add(fileName, ref FP);

                return true;

            }
            public void GetStat(string fileName)
            {

                if (fileName == null)
                    return;

                if (!Path.GetExtension(fileName).Equals(".mp3"))
                    return;



                try
                {

                    using (var reader = new Mp3FileReader(fileName))
                    {
                        var pcmLength = (int)reader.Length;
                        _leftBuffer = new byte[pcmLength / 2];
                        var buffer = new byte[pcmLength];
                        var bytesRead = reader.Read(buffer, 0, pcmLength);

                        int index = 0;
                        for (int i = 0; i < bytesRead; i += 4)
                        {
                            _leftBuffer[index] = buffer[i];
                            index++;
                            _leftBuffer[index] = buffer[i + 1];
                            index++;
                        }

                    }
                }
                catch (Exception exception)
                {
                    Console.WriteLine(exception.Message);
                    return;
                }
```

```
            var waveBuffer = new WaveBuffer(_leftBuffer);


            FourierTransformer ft = new FourierTransformer(4096);//разрешение по
частоте 32768 надо == 0.74 с


            double [][] data = new double[2048][];
            for(int i=0; i<2048; i++)
            {
                int k = _leftBuffer.Length / (2*2048) - 1;
                data[i] = new double[k];
            }

            bool flag = true;
            double max = 0;
            int freq, time;

            for (int i = 0; i <( _leftBuffer.Length / (2*2048) - 1); i++)
            {

                Complex[] data_interval = new Complex[4096];

                for (int k = 0; k < 4096; k++)
                {
                    if ((i * 2048 + 4096)>= _leftBuffer.Length / 2) //50%перекрытие
                    {
                        flag = false;
                        break;
                    }

                    double G = Gausse(k,4096); // Функция окна Гаусса

                    Complex comp = new Complex(waveBuffer.ShortBuffer[i * 2048 + k] *
G, 0); // G - для окна Гаусса
                    //Complex comp = new Complex(waveBuffer.ShortBuffer[i * 2048 +
k], 0);


                    data_interval[k] = comp;



                }
                // i параметр время ; k - частота
                if (flag)
                {
                    Complex[] xt = ft.Transform(data_interval);


                    int k = 0;

                    foreach (Complex c in xt)
                    {
                        if (k == 2048) break;

                        data[k][i] = Math.Pow(MoreMath.Sqr(c.Re) +
MoreMath.Sqr(c.Im), 0.5)/2048; // 20log(y)
```

```
                    k++;
                }


            }
        }


        bool corret = statForSample(ref data, this.pathDatabase, fileName);



    }

    static public double Gausse(double i, int framesize) // длина 4096
    {
        double a = (framesize - 1.0 ) / 2;
        double t = (i - a) / (0.5 * a);
        t = t * t;

        return Math.Exp(-t / 2);
    }
  }
}


//***************************************

// CBD.cs

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.SQLite;
using System.Data.Common;
using System.Data;
using System.IO;
using System.Runtime.InteropServices;

namespace AudioAnalyzer
{
    class CBD
    {

        private string Database;

        public CBD(string databaseName)
        {
            Database = databaseName;
            if (!File.Exists(databaseName))
            {
                SQLiteConnection.CreateFile(databaseName);
            }
        }

        //получить количество треков
        public int getCountTrack()
```

```
        {
            int result = 0;
            try
            {
                using (SQLiteConnection connection = new
SQLiteConnection(string.Format("Data Source={0};", Database)))
                {
                    connection.Open();
                    using (SQLiteCommand command = new SQLiteCommand(connection))
                    {
                        command.CommandText = "SELECT COUNT(*) FROM 'MUSIC';";

                        result = Convert.ToInt32(command.ExecuteScalar());

                    }
                }
            }
            catch (SQLiteException ex)
            {

            }
            return result;
        }
        public Music getTrackFromID( long id = 1) // получить трек начиная с id ,
если music.id == 0, то нет ответа
        {
            Music music = new Music();
            music.metrics = new double[10240];
            music.id = 0;

            try
            {
                using (SQLiteConnection connection = new
SQLiteConnection(string.Format("Data Source={0};", Database)))
                {
                    connection.Open();
                    using (SQLiteCommand command = new SQLiteCommand(connection))
                    {

                        command.CommandText = "SELECT * FROM 'MUSIC' ORDER BY 'id'
ASC;";

                        SQLiteDataReader reader = command.ExecuteReader();
                        foreach (DbDataRecord record in reader)
                        {
                            if ( (long)record["id"] < id) continue;
                            if ((long)record["id"] > id) return music;
                            music.id = (long)record["id"];
                            music.name = record["track"].ToString();

                            byte[] array = (byte[])record["param"];
                            FreqPacket fp = getPacket(ref array);
                            int k = 0;
                            foreach ( double stat in fp.data)
                            {
                                music.metrics[k] = stat;
                                k++;
                            }
                            music.mark = (long)record["mark"];

                            break;
```

```
                    }
                }
                connection.Close();
            }
        }
        catch (SQLiteException ex)
        {

        }

        return music;
    }

    public static byte[] getBytes(ref FreqPacket FP)
    {
        int size = Marshal.SizeOf(FP);
        byte[] arr = new byte[size];

        IntPtr ptr = Marshal.AllocHGlobal(size);
        Marshal.StructureToPtr(FP, ptr, true);
        Marshal.Copy(ptr, arr, 0, size);
        Marshal.FreeHGlobal(ptr);

        return arr;
    }

    public static FreqPacket getPacket(ref byte[] arrayF)
    {
        FreqPacket fp = new FreqPacket();

        int size = Marshal.SizeOf(fp);
        IntPtr ptr = Marshal.AllocHGlobal(size);

        Marshal.Copy(arrayF, 0, ptr, size);

        fp = (FreqPacket)Marshal.PtrToStructure(ptr, fp.GetType());
        //попробывать в строку
        Marshal.FreeHGlobal(ptr);

        return fp;
    }


    public void add(string trackName, ref FreqPacket arrayF)
    {
        byte[] array = getBytes(ref arrayF);
        int mark = 0; // 0 - не определена цена песни


        DirectoryInfo d = new DirectoryInfo(Path.GetDirectoryName(trackName));
        string markString = d.Name;

        if (!Int32.TryParse(markString, out mark))
            mark = 2; // цена не определена, то 2

        // Name directory 4 - mark = 4, e.t.c.
        //mark--; // в Python оценки с нуля - оценки 0-3

        try
        {
```

```
                using (SQLiteConnection connection = new
SQLiteConnection(string.Format("Data Source={0};", Database)))
                {
                    connection.Open();

                    using (SQLiteTransaction transaction =
connection.BeginTransaction())
                    {
                        using (SQLiteCommand command = new SQLiteCommand("CREATE
TABLE IF NOT EXISTS 'MUSIC' (id INTEGER PRIMARY KEY AUTOINCREMENT, track TEXT, param
BLOB, mark INTEGER);", connection))
                        {

                          command.ExecuteNonQuery();

                            command.CommandText = "INSERT INTO 'MUSIC'
('track','param','mark') VALUES (@trackName,@array,@mark)";
                            command.Parameters.Add("@array", DbType.Binary,
array.Length).Value = array;
                            command.Parameters.Add("@trackName", DbType.String,
trackName.Length * sizeof(char)).Value = trackName;
                            command.Parameters.Add("@mark", DbType.Int32,
sizeof(int)).Value = mark;
                            command.ExecuteNonQuery();
                        }

                        transaction.Commit();
                    }

                    connection.Close();
                }

            }
            catch (SQLiteException ex)
            {
              if (ex.ResultCode.Equals(SQLiteErrorCode.Busy))
                    Console.WriteLine("Database is locked by another process!");
            }
        }

    }
}

//*****************************************

// Statistic.cs

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Meta.Numerics;

namespace AudioAnalyzer
{
    class Statistic
    {
        int size;
        string databaseName;
```

```
        public Statistic(int sizeFrame, string dbPath)
        {
            size = sizeFrame;
            databaseName = dbPath;
        }
        public double median(ref double[][] frame, int index)
        {
            Array.Sort(frame[index]);
            // frame.length - четное
            return (frame[index][size/2] + frame[index][size/2 - 1])/2;
        }
        public double average(ref double[][] frame, int index)
        {

            return frame[index].Average();
        }

        public double stdev(ref double[][] frame, int index, double average)
        {
            double Sum = 0;
            double n = frame[index].Length - 1;

            foreach (double x in frame[index])
            {
                double k = MoreMath.Sqr(x - average)/n;
                Sum += k;
            }


            return Math.Pow(Sum,0.5);
        }

        public double skewness(ref double[][] frame, int index, double stdev, double
average)
        {
            double Sum = 0;
            double n = frame[index].Length;

            foreach ( double x in frame[index])
            {
                Sum += Math.Pow(((x - average)/stdev), 3);
            }


            return Sum/n;
        }

        public double kurtosis(ref double[][] frame, int index , double stdev, double
average)
        {
            double Sum = 0;
            double n = frame[index].Length;

            foreach (double x in frame[index])
            {
                Sum += Math.Pow(((x - average)/stdev), 4);
            }

            return Sum/n;
        }
```

```
        public static bool prescaler( ref double[] source, out double[] dest, int
scala = 10)
        {
            dest = new double[2000/scala*5];
            for ( int i=0; i<2000/scala; i++)
            {
                double s1, s2, s3, s4, s5;
                s1 = s2 = s3 = s4 = s5 = 0;
                for ( int g=0; g<scala; g++)
                {
                    s1 += source[i * 5 * scala + g * 5];
                    s2 += source[i * 5 * scala + g * 5 + 1];
                    s3 += source[i * 5 * scala + g * 5 + 2];
                    s4 += source[i * 5 * scala + g * 5 + 3];
                    s5 += source[i * 5 * scala + g * 5 + 4];
                }
                dest[i * 5] = s1 / scala;
                dest[i * 5 + 1] = s2 / scala;
                dest[i * 5 + 2] = s3 / scala;
                dest[i * 5 + 3] = s4 / scala;
                dest[i * 5 + 4] = s5 / scala;
            }

            return true;
        }



    }

}

//************************************

//IrPyBinding.cs

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using IronPython.Hosting;
using Microsoft.Scripting.Hosting;
using System.Diagnostics;
using System.IO;

namespace AudioAnalyzer
{
    class IrPyBinding
    {
        ScriptEngine engine;
        ScriptScope scope;

        string databaseName;
        string pathPython;

        public IrPyBinding( string DB)
        {
            engine = Python.CreateEngine();
```

```
            var paths = engine.GetSearchPaths();
            paths.Add(@"C:\Users\DEXP G115\Documents\Visual Studio
2015\Projects\AudioAnalyzer\AudioAnalyzer\Lib");
            engine.SetSearchPaths(paths);
            scope = engine.CreateScope();
            databaseName = DB;

        }

        public bool startTeachingPython()
        {
            return true;
        }

        public bool startEducate()
        {
            Process p = new Process();
            startPython(@"NeuroNet.py", p);

            return true;
        }

        public bool sendValidate()
        {

            Process p = new Process();
            startPython(@"NeuroReceiverValidate.py", p);

            CBD cbd = new CBD(databaseName);
            Music music;
            int result = cbd.getCountTrack();

            sendSize(result, ref engine, ref scope);

            for (int i = 1; i <= result; i++)
            {
                music = cbd.getTrackFromID(i);!= 0
                double[] message;

                Statistic.prescaler(ref music.metrics, out message);


                sendVector(ref message, music.mark, ref engine, ref scope);

            }



            return true;
        }

        public bool sendAllData()
        {

            Process p = new Process();
            startPython(@"NeuroReceiver.py", p);

            CBD cbd = new CBD(databaseName);
            Music music;
            int result = cbd.getCountTrack();
```

```
            sendSize( result, ref engine, ref scope);


            for (int i = 1; i <= result; i++)
            {
                music = cbd.getTrackFromID(i);
                double[] message;

                Statistic.prescaler(ref music.metrics, out message);


                sendVector(ref message, music.mark, ref engine, ref scope);

            }



            return true;
        }
        public static bool sendVector( ref  double[] data, long mark,ref ScriptEngine
engine, ref ScriptScope scope)
        {
            try
            {

                scope.SetVariable("doubleArray", data);
                scope.SetVariable("mark", (double)mark);

                engine.ExecuteFile(@"C:\Users\DEXP G115\documents\visual studio
2015\Projects\AudioAnalyzer\IronPythonNeuroNet\IronPythonBinding.py", scope);
            }
            catch (Exception exception)
            {
                Console.WriteLine(exception.Message);

            }

            return true;
        }

        public static bool sendSize( int size, ref ScriptEngine engine, ref
ScriptScope scope)
        {
            try
            {
                scope.SetVariable("size", size);
                engine.ExecuteFile(@"C:\Users\DEXP G115\documents\visual studio
2015\Projects\AudioAnalyzer\IronPythonNeuroNet\IPySendSize.py", scope);
            }
            catch (Exception exception)
            {
                Console.WriteLine(exception.Message);

            }
            return true;
        }

        public static void startPython( string filePython, Process p )
        {
```

```
            p.StartInfo = new ProcessStartInfo()
            {
                FileName = @"C:\Python27\python.exe",
                Arguments = filePython,
                WorkingDirectory = AppDomain.CurrentDomain.BaseDirectory,
                RedirectStandardOutput = false,
                UseShellExecute = false,
                CreateNoWindow = false,
                //Verb = "runas",
                RedirectStandardInput = false,
                RedirectStandardError = false
            };

            p.Start();


        }

    }
}

//************************************

//IrPythonSendSize.py

import socket
import cPickle

size


sock = socket.socket( socket.AF_INET, socket.SOCK_STREAM )

sock.connect(('localhost', 9090))

sock.send(cPickle.dumps(len(cPickle.dumps(size))))
data = cPickle.dumps(size)
sock.send(data)

sock.close()


print ("Bye from sendsize script")

//************************************************

//IrPythonBinding.py

import socket
#import pickle
import cPickle
#import System



print ("Hello")


doubleArray # array for initialize
mark # cost of track
```

```
#print(doubleArray[10])

sock = socket.socket( socket.AF_INET, socket.SOCK_STREAM )
sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

sock.connect(('localhost', 9090))


sock.send(cPickle.dumps(int(1000)))
k = 0

for digit in doubleArray:
    k = k + 1
    print k, digit
    sock.send(cPickle.dumps(len(cPickle.dumps(digit))))
    answer = sock.recv(2)
    data = cPickle.dumps(digit)
    sock.send(data)
    answer = sock.recv(2)

#send mar
sock.send(cPickle.dumps(len(cPickle.dumps(mark))))
print len(cPickle.dumps(mark))
answer = sock.recv(2)
data = cPickle.dumps(mark)
sock.send(data)
answer = sock.recv(2)
#answer = sock.recv(5)
sock.send("END")
sock.close()

#print answer
print ("Bye")


//*****************************************************************

//NeuroNet.py

from pybrain.tools.shortcuts import buildNetwork

from pybrain.structure import TanhLayer, SoftmaxLayer

from pybrain.supervised.trainers import BackpropTrainer

from pybrain.datasets import ClassificationDataSet

from sklearn.model_selection import train_test_split

import numpy as np

import matplotlib.pyplot as plt

import pickle

from pybrain.utilities import percentError
```

```
# should do function normalize before train
# get max and min for input vector
def loadDataFromCSV(X,Y):



    tf = open('metrics.csv','r')


    for line in tf.readlines():
        data = [float(x) for x in line.strip().split(',') if x != '']
        indata =  tuple(data[:1000])
        outdata = tuple(data[1000:])
        X.append(indata)
        Y.append(outdata)


        #data.addSample(indata,outdata)


    return


def Normilize(X,Xmax,Xmin):


    for i in range(len(X)):
        X[i]=(X[i] - Xmin)/(Xmax - Xmin)


    return


def buildDataSet(X,Y): # see example
```

```
    data = ClassificationDataSet(1000, 1, nb_classes=4, class_labels=['D', 'C', 'B', 'A'])


    for x,y in zip(X,Y):
        data.addSample(x, y)


    return data



print ("Hello")


Xmin, Xmax = 0,0
dataSize = 0
X = []
Y = []


loadDataFromCSV(X,Y)
Xmin,Xmax = np.min(X),np.max(X)
dataSize = np.size(Y)
Normilize(X, Xmax, Xmin)


print Xmax,Xmin
#print np.max(X),np.min(X)
print dataSize


X = np.array(X).reshape(dataSize,1000)
Y = np.array(Y).reshape(dataSize,1)


X_train, X_test, Y_train, Y_test = train_test_split(X, Y)
```

```
train = buildDataSet(X_train,Y_train)

test = buildDataSet(X_test, Y_test)

train._convertToOneOfMany()

test._convertToOneOfMany()

trainall = buildDataSet(X, Y)

trainall._convertToOneOfMany()



#print train

#print test



net = buildNetwork(1000, 500, 4, hiddenclass=TanhLayer, outclass=SoftmaxLayer)


#print net

maxepochs = len(X_train)*3

print maxepochs

results=[]

tests=[]


trainer = BackpropTrainer(net, train, weightdecay=0.0001, verbose=True) # train = dataset

plt.ion()

# test for calculate error in end

for i in range(maxepochs):

    aux = trainer.train()

    results.append(aux)
```

```
    x = np.arange(len(results))

    result_on_test = trainer.module.activateOnDataset(test)
    classes_predicted = np.argmax(result_on_test, axis=1)
    percError = percentError(classes_predicted, Y_test)
    tests.append(percError)
    xtest = np.arange(len(tests))

    plt.figure(1)
    plt.subplot(211)
    plt.plot(x, results,'ro',x, results)

    plt.subplot(212)
    plt.plot(xtest,tests,'ro',xtest,tests)

    plt.pause(0.05)

#print trainer.module.activate(X[0])



fileObject = open('net', 'w')

pickle.dump(net, fileObject)

fileObject.close()

#fileObject = open('net','r')
#net = pickle.load(fileObject)
```

```
#trainer1 = BackpropTrainer(net, train, weightdecay=0.0001, verbose=True)

#print trainer.module.activate(X[0])

#print net

# return error on test on socket

while True:

    plt.pause(0.05)

#trainer.trainUntilConvergence(maxEpochs=100)


#print trainer.module.activate(X[0])


print ("END")

//*****************************************
```

//NeuroReceiver.py

```
#import pickle

from socket import *

import cPickle, sys, socket, clr, pprint, csv

from encodings.punycode import digits

#import scipy

#import numpy

#import matplotlib.pyplot




print ("Hello")
```

```
sock = socket.socket( socket.AF_INET, socket.SOCK_STREAM)
#sock.setsockopt(SOL_SOCKET, SO_REUSEADDR, 1)
sock.bind(('', 9090))
sock.listen(1)
conn, addr = sock.accept()

print 'connected:', addr

size = conn.recv(10)
kol = conn.recv(cPickle.loads(size))
k = 0
conn.close()




while k < cPickle.loads(kol):

    # new connection with other script
    sock.listen(1)
    conn, addr = sock.accept()
    #conn.settimeout(60)

    print 'connected:', addr

    lenarray = conn.recv(7) # length massiv
```

```
#conn.send("OK")

k = k + 1

array_size = cPickle.loads(lenarray)

print array_size

i = 0

row = []


while i < array_size :

   lendigit = conn.recv(30)

   conn.send("OK")

   data = conn.recv(cPickle.loads(lendigit))

   #data = conn.recv(30)

   digit = cPickle.loads( data )


   print k,": ",i,". ",digit

   row.append(digit)

   i = i + 1

   conn.send("OK")


# receive mark

lenmark = conn.recv(10)# receive digit from 1 to 4

conn.send("OK")

data = conn.recv(cPickle.loads(lenmark))

mark = cPickle.loads(data)

row.append(mark)

conn.send("OK")

print mark
```

```
    #conn.send("OK")
    answer = conn.recv(3)
    print answer
    conn.close()

    with open('metrics.csv', 'ab') as csv_file:
        csv_writer = csv.writer(csv_file)
        csv.writer(csv_file,  dialect='excel' ).writerow(row)

#pprint.pprint( data )



#print data_array + 1


#print repr(data_array)


//*************************************************************
// NeuroValidate.py
from pybrain.datasets import ClassificationDataSet
import numpy as np
import pickle
from pybrain.tools.shortcuts import buildNetwork
from pybrain.structure import TanhLayer, SoftmaxLayer
from pybrain.supervised.trainers import BackpropTrainer
import csv


def loadDataFromCSV(X,Y):
```

```
    tf = open('validate.csv','r')

    for line in tf.readlines():
        data = [float(x) for x in line.strip().split(',') if x != '']
        indata =  tuple(data[:1000])
        outdata = tuple(data[1000:])
        X.append(indata)
        Y.append(outdata)

        #data.addSample(indata,outdata)

    return

def Normilize(X,Xmax,Xmin):

    for i in range(len(X)):
        X[i]=(X[i] - Xmin)/(Xmax - Xmin)

    return

def buildDataSet(X,Y): # see example
    data = ClassificationDataSet(1000, 1, nb_classes=4, class_labels=['D', 'C', 'B', 'A'])

    for x,y in zip(X,Y):
        data.addSample(x, y)

    return data
```

```
Xmin, Xmax = 0,0

dataSize = 0

X = []

Y = []


loadDataFromCSV(X,Y)

Xmin,Xmax = np.min(X),np.max(X)

dataSize = np.size(Y)

Normilize(X, Xmax, Xmin)


X = np.array(X).reshape(dataSize,1000)

Y = np.array(Y).reshape(dataSize,1)


test = buildDataSet(X, Y)

test._convertToOneOfMany()


fileObject = open('net','r')

net = pickle.load(fileObject)


trainer = BackpropTrainer(net, test, weightdecay=0.0001, verbose=True)


result_on_test = trainer.module.activateOnDataset(test)


print result_on_test


# matrix to file
```

```
for i in range(len(result_on_test)):
    with open('resultValidate.csv','ab') as csv_file:
        csv_writer = csv.writer(csv_file)
        csv.writer(csv_file,  dialect='excel' ).writerow(result_on_test[i])
```