ChatGPT

Курс: Классификация жанра аудио и рекомендательная система на Python

Часть 1: Классификация жанра аудиофайла

1. Введение в нейронные сети

Искусственная нейросеть (ИНС) моделируется как совокупность связанных узлов (нейронов), повторяющих упрощённую схему биологического нейрона [1]. Каждый нейрон получает на вход несколько сигналов, вычисляет их взвешенную сумму и пропускает результат через **активационную функцию** (например, сигмоида или ReLU) [1]. Искусственные нейроны организованы в **слои**: входной, один или несколько скрытых и выходной [2]. На входной слой подаётся исходный сигнал (например, признаки аудио), скрытые слои выполняют нелинейную обработку, а выходной выдаёт предсказание (например, жанр песни).

Обучение нейросети заключается в автоматическом подборе весов связей между нейронами. Сеть **не программируется** вручную, она **обучается** на размеченных данных (метках классов) [3]. Во время обучения веса настраиваются алгоритмом обратного распространения ошибки (бэктпроп), минимизирующим функцию потерь. В результате успешно обученная сеть выявляет сложные зависимости между входными признаками и классами и может обобщать свой опыт на новые данные [3].

2. Установка необходимых библиотек

Для обработки аудиосигналов и создания модели на Python понадобятся библиотеки:

- **librosa** — для загрузки и анализа аудио;
- **TensorFlow (Keras) или PyTorch** — для построения и обучения нейросети;
- **numpy, pandas, sklearn** — для работы с данными и подготовки признаков.

Обычно библиотеки устанавливают через `pip`:

```
pip install librosa tensorflow numpy pandas scikit-learn spotipy
```

Библиотека *librosa* — это пакет Python для анализа музыки и аудио, он предоставляет инструменты для извлечения музыкальных признаков [4]. Например, так загружают аудиофайл:

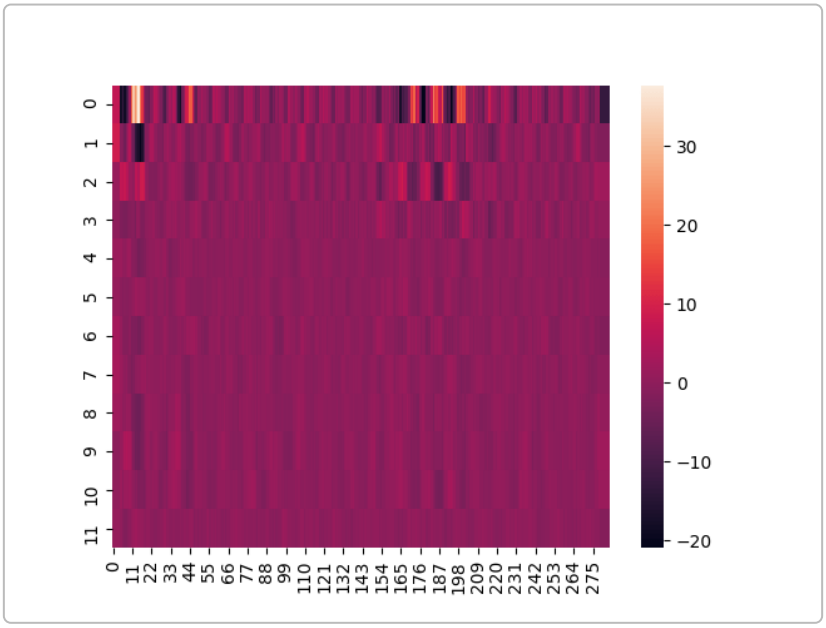
```
import librosa

# Загрузка mp3 (можно и другие форматы).
# sr=22050 задаёт частоту дискретизации 22050 Гц.
signal, sr = librosa.load('path/to/file.mp3', sr=22050)
print(signal.shape, sr)  # (число семплов,), 22050
```

После загрузки `signal` — это одномерный NumPy-массив (временной ряд амплитуд), а `sr` — частота дискретизации. По умолчанию звук конвертируется в моно.

3. Извлечение признаков из аудио

Для классификации жанра raw-аудио следует преобразовать в числовые признаки. Одним из ключевых признаков являются **мел-кепстральные коэффициенты (MFCC)** [5] [6]. MFCC отражают тембровые характеристики звука (например, разницу между «металлической» и «деревянной» гитарой) и широко используются в задачах аудиоанализа [6].



Ниже код на Python для извлечения MFCC (и других признаков) с помощью librosa. Комментарии поясняют каждый шаг:

```
import numpy as np

# Предполагаем, что signal и sr получены через librosa.load (см. предыдущий раздел).
# Вычисление MFCC. n_mfcc=40 означает, что получаем 40 коэффициентов MFCC на фрейм.
mfcc = librosa.feature.mfcc(y=signal, sr=sr, n_mfcc=40)
# mfcc.shape = (40, T), где T зависит от длины аудио и hop_length.

# Усредняем MFCC по времени, чтобы получить фиксированный вектор признаков из MFCC.
mfcc_mean = np.mean(mfcc, axis=1)  # Размерность (40,)
```

MFCC получается в виде матрицы (количество коэффициентов × количество фреймов). Здесь мы усредняем по времени, чтобы получить из MP3 одного трека вектор длины 40, пригодный для классификации. Кроме MFCC можно добавить и другие признаки (спектральный центроид, спектральный распад и т.д.), но мы ограничимся MFCC.

4. Подготовка датасета

Популярным датасетом для классификации музыкальных жанров является **GTZAN**. Это 1000 аудиофайлов по 10 жанрам (100 файлов на жанр) [7]. Каждый трек длится 30 секунд; жанры включают блюз, классику, джаз, рок, металл и т.д. Например, воспользуемся библиотекой `datasets` (Hugging Face) для загрузки GTZAN:

```
from datasets import load_dataset

# Загружаем датасет GTZAN (999 вместо 1000 треков, т.к. один повреждён) [7].
gtzan = load_dataset("marsyas/gtzan", "all")
print(gtzan)
# Разбиваем на обучающую и тестовую выборки (например, 90%/10%).
gtzan = gtzan['train'].train_test_split(test_size=0.1, seed=42)
```

Этот код создаст словарь с разделами `train` и `test` с соответствующими аудиомассивами и метками жанров (как целые числа). Также можно самостоятельно скачать GTZAN (например, с Kaggle или официального сайта) и разложить по папкам жанров. Важно, чтобы аудио было в одном формате и частоте (например, все 22050 Гц).

После загрузки данных пробегаем по файлам, извлекаем признаки MFCC (как выше) и собираем два массива: `X` (признаки) и `y` (метки жанров). Затем нормализуем или стандартизируем признаки (например, с помощью `sklearn.preprocessing.StandardScaler`) и разделяем на обучающую/тестовую выборки.

5. Построение и обучение модели классификации

Построим простую полносвязную сеть (MLP) для классификации жанра. Используем TensorFlow Keras:

```
import tensorflow as tf
from tensorflow import keras

num_classes = 10  # жанров в GTZAN
input_dim = mfcc_mean.shape[0]  # например, 40

model = keras.Sequential([
    keras.layers.Input(shape=(input_dim,)),
    keras.layers.Dense(128, activation='relu'),    # первый скрытый слой
    keras.layers.Dense(64, activation='relu'),     # второй скрытый слой
    keras.layers.Dense(num_classes, activation='softmax')  # выходной слой
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
print(model.summary())
```

В этом примере последний слой имеет `softmax`-активацию, т.к. это многоклассовая классификация (10 жанров). Используем `sparse_categorical_crossentropy`, потому что метки представлены числами от 0 до 9. Затем запускаем обучение:

```
# X_train и y_train должны быть подготовлены заранее.
model.fit(X_train, y_train, epochs=20, batch_size=16, validation_split=0.1)
```

В комментариях кода можно подробно описать каждый шаг: определение слоёв, выбор активаций (`relu` в скрытых слоях для нелинейности, `softmax` для выхода), компиляция модели и запуск обучения. В конце мы оцениваем точность на тестовой выборке:

```
loss, acc = model.evaluate(X_test, y_test)
print(f"Test accuracy: {acc:.2f}")
```

Таким образом мы обучили нейросеть, которая по MFCC аудиофайла определяет жанр.

Часть 2: Рекомендательная система

6. Формулировка задачи и архитектура

Вторая модель — рекомендательная система, учитывающая жанр (из модели 1), язык песни и диапазон годов. На входе пользователь указывает, например, список уже имеющихся треков (их жанры), предпочтительный язык и интервал выпуска (например, 1990–2000). Задача — предложить похожие треки.

Проще всего сделать **контентную рекомендацию**: сопоставлять треки по близости их признаков (жанр, язык, год, аудио-признаки). Для примера заведение небольшой «базы» треков со структурой:

```
tracks = [
    {"name": "Song A", "genre": "rock", "language": "English", "year": 1995},
    {"name": "Song B", "genre": "rock", "language": "English", "year": 1998},
    {"name": "Song C", "genre": "pop",  "language": "Russian", "year": 2010},
    # ... и т.д.
]
```

Можно представить признаки каждого трека в виде вектора (например, one-hot кодирование жанра и языка + нормированное число года) и затем подобрать ближайшие по Евклиду или косинусной мере к «профилю пользователя» (который можно получить усреднением векторов уже имеющихся треков). Это простой подход без «обучения» (скорее алгоритмический).

Однако многие сервисы используют **коллаборативную фильтрацию**: находят пользователей со сходными вкусами и рекомендуют треки из их прослушанного [8]. Для нашей задачи (учёт конкретных жанров и годов) проще использовать фильтры. Например: 1. Отфильтровать все доступные треки по указанному языку и диапазону годов.
2. Из этой подвыборки выбрать те, чей жанр совпадает с жанрами треков пользователя или близок к ним.

Можно ещё расширить модель: например, обучить простую нейросеть, которая по входным признакам (жанры, язык, год) предсказывает вероятность интереса пользователя к каждому треку (что сложно на практике), но для курса достаточно алгоритмической реализации.

7. Интеграция с внешним API (Spotify)

Для демонстрации можно воспользоваться Spotify Web API через библиотеку Spotipy [9]. Spotipy позволяет получать данные о треках, плейлистах и делать рекомендации. Сначала регистрируем приложение на Spotify Developer Dashboard и получаем `client_id` и `client_secret`, затем:

```
import spotipy
from spotipy.oauth2 import SpotifyClientCredentials

# Установите CLIENT_ID и CLIENT_SECRET, полученные на Spotify for Developers
client_id = 'ВАШ_CLIENT_ID'
client_secret = 'ВАШ_CLIENT_SECRET'

auth_manager = SpotifyClientCredentials(client_id=client_id,
client_secret=client_secret)
sp = spotipy.Spotify(auth_manager=auth_manager)

# Пример: получить рекомендации по одному жанру
results = sp.recommendations(seed_genres=['rock'], limit=5)
for track in results['tracks']:
    print(track['name'], '-', track['artists'][0]['name'])
```

Функция `Spotify.recommendations()` позволяет получить список похожих треков по «зернам» (жанрам, исполнителям или трекам) [10]. Мы можем передать жанры из первой модели и, например, жанры и треки пользователя.

Кроме того, в Spotify можно делать поиск с фильтрами по жанру и году. В запросе к `sp.search()` можно указать параметры `genre:` и `year:`. Например:

```
genre = 'rock'
year_min, year_max = 1990, 2000
query = f'genre:{genre} year:{year_min}-{year_max}'
results = sp.search(q=query, type='track', limit=5)
for item in results['tracks']['items']:
    name = item['name']
    artist = item['artists'][0]['name']
    release_date = item['album']['release_date']
    print(f"{name} - {artist} ({release_date})")
```

Это вернёт треки указанного жанра из нужного диапазона годов. Фильтрация по языку в Spotify напрямую не поддерживается, но можно ограничиться жанром и годом, а язык учитывать, например, по выбору самих треков (или по метаданным).

8. Пример простой рекомендательной системы на Python

Объединим всё: допустим, у нас есть предсказанные жанры из модели 1 (`predicted_genre`), выбранный пользователем язык (`lang`) и годовой диапазон (`y1, y2`). Для простоты реализуем фильтрацию по трекам из нашей «базы»:

```
def recommend_tracks(user_genres, language, year_range, catalog):
    """
    user_genres: список жанров (например, ['rock', 'pop'])
    language: строка, напр. 'English'
    year_range: кортеж (год_мин, год_макс)
    catalog: список словарей треков (name, genre, language, year)
    """
    y_min, y_max = year_range
    # Фильтрация по языку и диапазону годов
    candidates = [track for track in catalog
                  if track['language'] == language
                  and y_min <= track['year'] <= y_max]
    # Оставляем те треки, жанр которых совпадает с одним из жанров
пользователя
    recs = [track for track in candidates if track['genre'] in user_genres]
    # Если рекомендаций мало, можно добавить похожие жанры или расширить
поиск
    return recs

# Пример использования:
catalog = [
    {"name": "Song A", "genre": "rock", "language": "English", "year": 1995},
    {"name": "Song B", "genre": "rock", "language": "English", "year": 1998},
    {"name": "Song C", "genre": "pop",  "language": "English", "year": 1992},
    {"name": "Song D", "genre": "pop",  "language": "Russian", "year": 1995},
    # ...
]
user_genres = ['rock']
recs = recommend_tracks(user_genres, 'English', (1990, 2000), catalog)
for t in recs:
    print("Recommended:", t['name'], "-", t['genre'])
```

Данный пример — очень упрощённая контентная рекомендация. Для улучшения можно использовать более сложные метрики сходства (например, косинус угла между векторами признаков треков) и учитывать не только совпадение жанра.

Для демонстрации подключения к внешнему API можно показать работу функции `sp.recommendations`, как выше. Таким образом курс даёт понятия, как получить вводные данные от пользователя (жанры, язык, годы), как отфильтровать и предложить похожие композиции.

Источники: определения и концепции нейросетей и признаков аудио взяты из научных и образовательных ресурсов [1] [2] [3] [4] [5]. Пример использования Spotify API основан на

официальной документации Spotipy [9] [10]. Рекомендательные системы часто применяют коллаборативную фильтрацию и анализ контента [8].

---

[1] Искусственный нейрон — Википедия
https://ru.wikipedia.org/wiki/%D0%98%D1%81%D0%BA%D1%83%D1%81%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BD%D0%B5%D0%B9%D

[2] Что такое искусственные нейронные сети? Виды, модели и задачи
https://cloud.ru/blog/neural-networks

[3] Нейронная сеть — Википедия
https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C

[4] [6] Библиотека Librosa Python для обработки и анализа аудиофайлов
https://pythonru.com/biblioteki/librosa

[5] Классификация аудиофайлов с библиотекой Librosa / Хабр
https://habr.com/ru/companies/otus/articles/741080/

[7] Дообучение модели для классификации музыки - Hugging Face Audio Course
https://huggingface.co/learn/audio-course/ru/chapter4/fine-tuning

[8] Как работают рекомендательные системы музыкальных сервисов | РБК Тренды
https://trends.rbc.ru/trends/industry/5f2c271c9a7947ead6c655f3

[9] [10] Welcome to Spotipy! — spotipy 2.0 documentation
https://spotipy.readthedocs.io/en/2.25.2/