

Использование библиотеки pyRiemann для классификации данных EEG (электроэнцефалографии)

Анишкова Анастасия Сергеевна

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

Цель этого исследования — применить библиотеку pyRiemann для анализа и классификации данных электроэнцефалографии (EEG). Работа выполнялась в облачной среде Google Colab с использованием языка Python, что обеспечило удобство и гибкость в проведении экспериментов. Библиотека pyRiemann - это мощный инструмент для обработки многоканальных сигналов EEG, который позволяет эффективно классифицировать данные на основе римановой геометрии. Такой подход полезен в различных областях, включая медицинскую диагностику (например, выявление эпилепсии или нарушений сна), нейрофидбек и разработку интерфейсов «мозг-компьютер». Исследование демонстрирует, как pyRiemann может повысить точность анализа EEG и стать основой для создания новых технологий в нейронауках и биомедицинской инженерии.

Ключевые слова: Google Colab, pyRiemann, анализа и классификации данных, EEG, Python.

Using the pyRiemann library for classifying EEG (Electroencephalography) data

Anishkova Anastasia Sergeevna

Sholom Aleichem Priamurskiy State University

Student

Abstract

The aim of this study is to apply the pyRiemann library for the analysis and classification of electroencephalography (EEG) data. The work was performed in the Google Colab cloud environment using Python, which provided convenience and flexibility in conducting experiments. The pyRiemann library is a powerful tool for processing multi-channel EEG signals, which allows efficient classification of data based on Riemannian geometry. This approach is useful in various fields, including medical diagnostics (for example, the detection of epilepsy or sleep disorders), neurofeedback, and the development of brain-computer interfaces. The study demonstrates how pyRiemann can improve the accuracy of EEG analysis and become the basis for the creation of new technologies in neuroscience and biomedical engineering.

Keywords: Google Colab, pyRiemann, Data analysis and Classification, EEG, Python.

1 Введение

1.1 Актуальность

Актуальность связана с возрастающим интересом к разработке технологий анализа мозговой активности и их применению в различных областях. Библиотека pyRiemann предоставляет современные методы обработки многоканальных сигналов на основе римановой геометрии, что делает её мощным инструментом для классификации данных EEG. Это особенно важно для развития интерфейсов «мозг-компьютер», которые позволяют управлять устройствами силой мысли, а также для медицинской диагностики, где точная классификация паттернов EEG помогает выявлять заболевания, такие как эпилепсия или нарушения сна. Кроме того, pyRiemann находит применение в нейрофидбеке, когнитивных исследованиях и разработке новых технологий, таких как умные протезы и системы управления. Открытость и доступность библиотеки способствуют её активному использованию в научных и прикладных исследованиях, что делает её ключевым инструментом в области анализа EEG и машинного обучения.

1.2 Обзор исследований

А.М. Самохина описала непрерывное представление времени в задачах декодирования сигналов [1], Применение римановой геометрии к классификации BCI продемонстрировал А. Барахант [2], М. Конгедо, А. Барачант, Р. Бхатия рассказали о римановой геометрии для интерфейсов мозг-компьютер на основе ЭЭГ [3], Классификации мультиклассовых интерфейсов мозг-компьютер по римановой геометрии охарактеризовал А. Барачант [4], Ф. Игер, М. Берар, Ф. Лотте рассмотрели Римановы подходы в интерфейсах мозг-компьютер [5].

1.3 Цель исследования

Цель исследования заключается в применении библиотеки pyRiemann для классификации данных электроэнцефалографии (EEG) с использованием современных методов анализа на основе римановой геометрии. Исследование направлено на демонстрацию возможностей pyRiemann для повышения точности и эффективности обработки многоканальных сигналов EEG

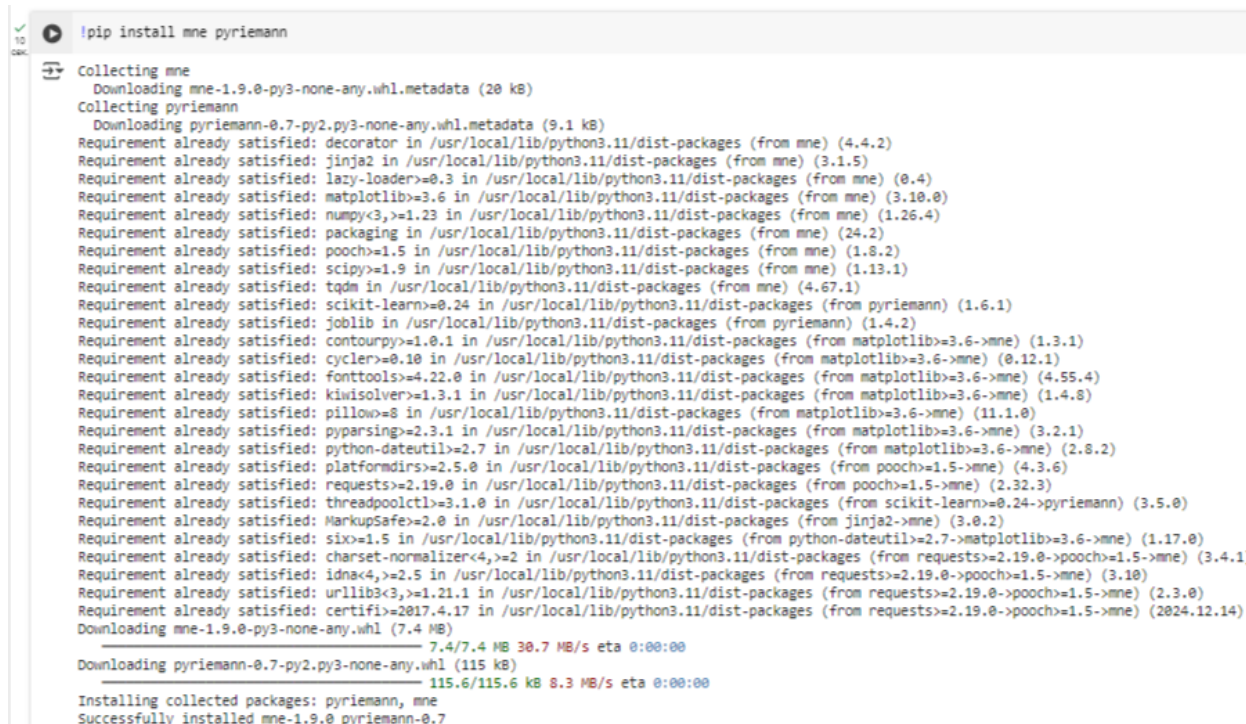
2 Материалы и методы

В данном исследовании используется среда Google Colab и язык программирования Python.

3 Результаты

Начнем с установки библиотеки pyRiemann (рис.1), которая является мощным инструментом для анализа многомерных данных, таких как EEG и

MEG, с применением римановой геометрии для работы с ковариационными матрицами.



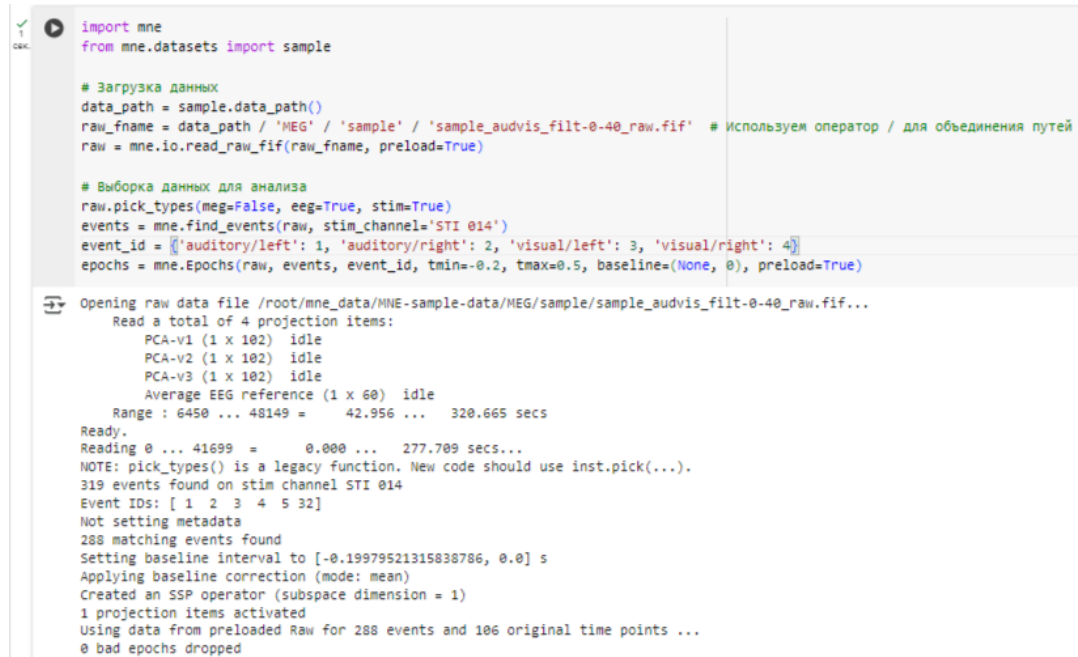
```
!pip install mne pyriemann

Collecting mne
  Downloading mne-1.9.0-py3-none-any.whl.metadata (20 kB)
Collecting pyriemann
  Downloading pyriemann-0.7-py2.py3-none-any.whl.metadata (9.1 kB)
Requirement already satisfied: decorator in /usr/local/lib/python3.11/dist-packages (from mne) (4.4.2)
Requirement already satisfied: Jinja2 in /usr/local/lib/python3.11/dist-packages (from mne) (3.1.5)
Requirement already satisfied: lazy-loader>=0.3 in /usr/local/lib/python3.11/dist-packages (from mne) (0.4)
Requirement already satisfied: matplotlib>=3.6 in /usr/local/lib/python3.11/dist-packages (from mne) (3.10.0)
Requirement already satisfied: numpy<3,>=1.23 in /usr/local/lib/python3.11/dist-packages (from mne) (1.26.4)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from mne) (24.2)
Requirement already satisfied: pooch>=1.5 in /usr/local/lib/python3.11/dist-packages (from mne) (1.8.2)
Requirement already satisfied: scipy>=1.9 in /usr/local/lib/python3.11/dist-packages (from mne) (1.13.1)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from mne) (4.67.1)
Requirement already satisfied: scikit-learn>=0.24 in /usr/local/lib/python3.11/dist-packages (from pyriemann) (1.6.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages (from pyriemann) (1.4.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.6->mne) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.6->mne) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.6->mne) (4.55.4)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.6->mne) (1.4.8)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.6->mne) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.6->mne) (3.2.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.6->mne) (2.8.2)
Requirement already satisfied: platformdirs>=2.5.0 in /usr/local/lib/python3.11/dist-packages (from pooch>=1.5->mne) (4.3.6)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.11/dist-packages (from pooch>=1.5->mne) (2.32.3)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=0.24->pyriemann) (3.5.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from Jinja2->mne) (3.0.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->matplotlib>=3.6->mne) (1.17.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests>=2.19.0->pooch>=1.5->mne) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests>=2.19.0->pooch>=1.5->mne) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests>=2.19.0->pooch>=1.5->mne) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests>=2.19.0->pooch>=1.5->mne) (2024.12.14)
Downloading mne-1.9.0-py3-none-any.whl (7.4 MB)
7.4/7.4 MB 30.7 MB/s eta 0:00:00
Downloading pyriemann-0.7-py2.py3-none-any.whl (115 kB)
115.6/115.6 kB 8.3 MB/s eta 0:00:00
Installing collected packages: pyriemann, mne
Successfully installed mne-1.9.0 pyriemann-0.7
```

Рисунок 1. Установка библиотеки

Подготовим данные для последующего анализа: загрузим данные EEG из файла в формате .fif, выберем EEG-каналы и стимуляционные каналы для анализа, найдем события (триггеры), соответствующие моментам предъявления стимулов, и создадим эпохи - отрезки данных, связанные с событиями, для дальнейшего анализа (рис.2).

Данные взяты из MNE Sample Dataset [1], который является частью библиотеки mne. Файл .fif - это стандартный формат для хранения нейрофизиологических данных, используемый в MEG/ EEG. В файле содержатся записи сигналов, включая MEG-каналы (магнитоэнцефалография), EEG-каналы (электроэнцефалография) и стимуляционные каналы (триггеры), а также метаданные: информация о каналах (названия, типы, расположение датчиков), частота дискретизации (в данном случае 600 Гц) и временные метки событий (триггеры).



```

import mne
from mne.datasets import sample

# Загрузка данных
data_path = sample.data_path()
raw_fname = data_path / 'MEG' / 'sample' / 'sample_audvis_filt-0-40_raw.fif' # Используем оператор / для объединения путей
raw = mne.io.read_raw_fif(raw_fname, preload=True)

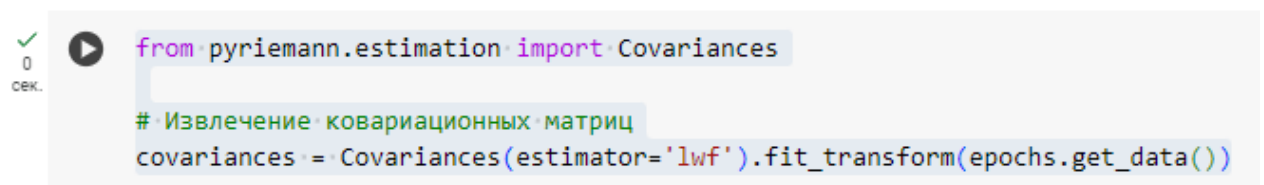
# Выборка данных для анализа
raw.pick_types(meg=False, eeg=True, stim=True)
events = mne.find_events(raw, stim_channel='STI 014')
event_id = {'auditory/left': 1, 'auditory/right': 2, 'visual/left': 3, 'visual/right': 4}
epochs = mne.Epochs(raw, events, event_id, tmin=-0.2, tmax=0.5, baseline=(None, 0), preload=True)

Opening raw data file /root/mne_data/MNE-sample-data/MEG/sample/sample_audvis_filt-0-40_raw.fif...
Read a total of 4 projection items:
  PCA-v1 (1 x 102) idle
  PCA-v2 (1 x 102) idle
  PCA-v3 (1 x 102) idle
  Average EEG reference (1 x 60) idle
Range : 6450 ... 48149 = 42.956 ... 320.665 secs
Ready.
Reading 0 ... 41699 = 0.000 ... 277.709 secs...
NOTE: pick_types() is a legacy function. New code should use inst.pick(...).
319 events found on stim channel STI 014
Event IDs: [ 1  2  3  4  5 32]
Not setting metadata
288 matching events found
Setting baseline interval to [-0.19979521315838786, 0.0] s
Applying baseline correction (mode: mean)
Created an SSP operator (subspace dimension = 1)
1 projection items activated
Using data from preloaded Raw for 288 events and 106 original time points ...
0 bad epochs dropped

```

Рисунок 2. Подготовка данных для анализа

Вычислим ковариационные матрицы для каждой эпохи данных EEG (рис.3), которые описывают взаимодействие между каналами и используются для последующего анализа, включая классификацию с применением римановой геометрии.



```

from pyriemann. estimation import Covariances

# Извлечение ковариационных матриц
covariances = Covariances(estimator='lwf').fit_transform(epochs.get_data())

```

Рисунок 3. Вычисление ковариационные матрицы

Разделим данные на обучающую и тестовую выборки, после чего классификатор MDM будет обучен на ковариационных матрицах обучающих данных, предскажет метки классов для тестовых данных, а затем оценит и выведет точность классификации (рис.4). Результат Accuracy: 94.83% означает, что классификатор MDM (Minimum Distance to Mean) правильно предсказал 94.83% меток классов на тестовой выборке. Это свидетельствует о том, что классификатор хорошо обучился на данных, ковариационные матрицы эффективно описывают различия между классами, а риманова геометрия подходит для анализа данных EEG в данном контексте.

```

from pyriemann.classification import MDM
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Разделение данных на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(covariances, epochs.events[:, 2], test_size=0.2, random_state=42)

# Обучение классификатора
mdm = MDM()
mdm.fit(X_train, y_train)

# Предсказание на тестовой выборке
y_pred = mdm.predict(X_test)

# Оценка точности
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')

```

Accuracy: 94.83%

Рисунок 4. Обучение выборки

Построим матрицу ошибок (confusion matrix), которая визуализирует результаты классификации (рис.5).

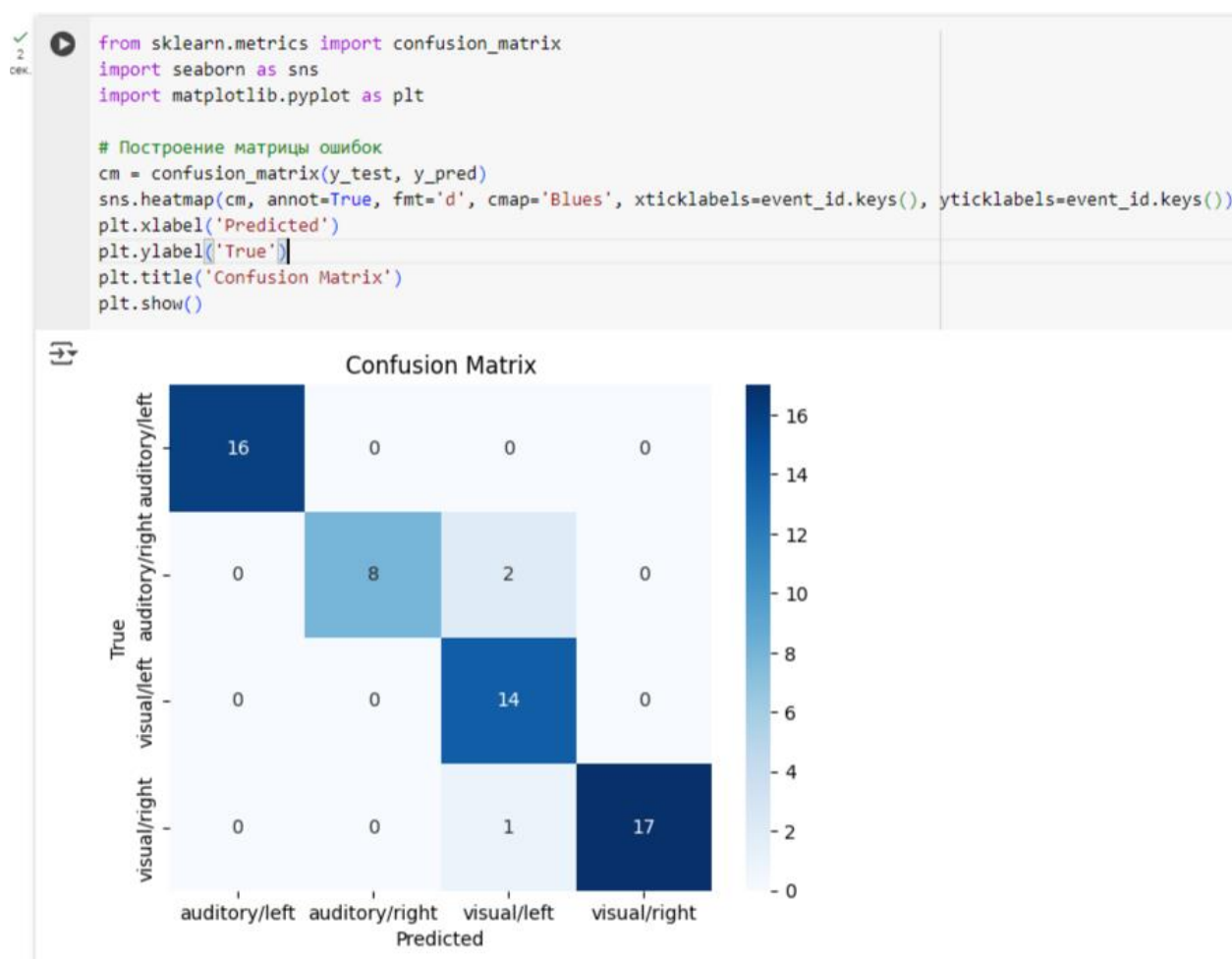


Рисунок 5. Матрица ошибок

Матрица ошибок показывает, что классификатор хорошо справляется с классами 1, 3 и 4, но допускает ошибки при классификации класса 2, путая его с классом 3. Это указывает на необходимость улучшения модели для работы с этими классами.

Библиографический список

1. URL:
https://mne.tools/stable/auto_tutorials/preprocessing/40_artifact_correction_ica.html (дата обращения: 27.01.2025);
2. Самохина А.М. Непрерывное представление времени в задачах декодирования сигналов: дис. маг-т Интеллектуальный анализ данных наук: 010990. Москва, 2021. 32 с.;
3. Barachant A. Riemannian geometry applied to BCI classification //International conference on latent variable analysis and signal separation. Berlin, Heidelberg: Springer Berlin Heidelberg. 2010. С. 629-636.;
4. Congedo M., Barachant A., Bhatia R. Riemannian geometry for (EEG)-based brain-computer interfaces; a primer and a review //Brain-Computer Interfaces. 2017. №. 3. С. 155-174.;
5. Barachant A. et al. Multiclass brain-computer interface classification by Riemannian geometry //IEEE Transactions on Biomedical Engineering. 2011. №. 4. С. 920-928.;
6. Yger F., Berar M., Lotte F. Riemannian approaches in brain-computer interfaces: a review //IEEE Transactions on Neural Systems and Rehabilitation Engineering. 2016. №. 10. С. 1753-1762.