pycaspule - Python package

Impulse Neiry

May 5, 2021

Contents

1 Introduction

We provide a `pycapsule` package with various tools for working with Capsule BCI system. At this moment it allows to:

1. Read session records into memory
2. Import data into MNE

1.1 Installation

Python package can be installed via pip:

```
pip install pycapsule
```

Alternatively it can be found at `{SDK-ROOT}/Tools/Python/Packages/pycapsule`.

2 Reading data with RecordReader

Record reader functionality can be used to unpack stream data from sessions ran by Capsule. Reader works with session data files (.dat/.rec) in pairs. Both are required in order to successfully unpack data. Unpacking data using our package requires implementing a subclass of `RecordReaderVisitor`. It is a class which will receive all incoming packets as they are unpacked.

- `P300ProcessingUnit` packet contains all the data necessary to perform model training and predictions.
- `Raw EEG` and `Raw Resistance` packets contain continuous signal data for the whole session.
- `Interface data` packets contain specific commands related to a particular interface, these commands are issued by client.

Let’s look at a simple example of how to unpack session data:

Listing 1: Usage example

```
1  from pycapsule.record_reader import *
2
3  class Visitor(RecordReaderVisitor):
4      def OnP300ProcessingUnit(self, p300unit:P300ProcessingUnit):
5          print("P300 Unit!")
6      def OnRawEEG(self, eegData:np.ndarray, eegTimestamps:np.array):
7          print("Raw EEG!")
8      def OnRawResistance(self, resData:np.ndarray):
9          print("Raw Resistance!")
10     def OnInterfaceData(self, interfaceData):
11         print("Interface Data!")
12
13 visitor = Visitor()
14
15 filepath = "D:/Capsule-recs/Impulse - 2020.12.07 - 07.56.32/session.rec"
16 RecordReader.Unpack(filepath, visitor)
```

We will now review steps required to retrieve data:

line 1 Import Capsule’s `record_reader` required for data unpacking

line 3 Implement subclass of a `RecordReaderVisitor`, there is no need need to implement every function

line 16 Call unpack functionality with a `filepath` to a session recording (.rec data file) and instance of a visitor we implemented

Here's a basic reference of what data classes contain:

Listing 2: Data classes

```
@dataclass
class P300StimulusData:
    stimulusId: int
    sampleStartIndex: int          # index into eegData
    timestamp: float               # in seconds

@dataclass
class P300ProcessingUnit:
    unitId: int
    actId: int
    targetStimulus: int
    stimuliCount: int
    eegData: np.ndarray            # shape(channels, samples)
    eegTimestamps: np.array        # shape(samples)
    stimuliData: typing.List[P300StimulusData]
    shouldEndLearn: bool           # (legacy) signals BCI system to train a model
```

`P300ProcessingUnit` in a basic sense represents a sequence of stimuli activations. Stimuli with their id, timestamp and index into `eegData` are stored in `stimuliData`. EEG data for a sequence is stored in `eegData` numpy array with shape (channels, samples).

3 Importing data into MNE

It is also possible to import records directly into MNE.

Listing 3: Usage example

```
from pycapsule.mne.io import read_raw_csr
import matplotlib

filepath = "D:/Capsule-recs/Impulse - 2020.12.07 - 07.56.32/session.rec"
raw, events, event_id = read_raw_csr(filepath)
raw.set_montage('standard_1020', match_case=False) # montage to visualize electrode placement
raw.filter(1, 40) # since data is not filtered, we apply basic filter
raw.plot(events, event_id=event_id, event_color={1: 'g', 2: 'b', 3: 'r'})
raw.plot_psd()
matplotlib.pyplot.show()
```

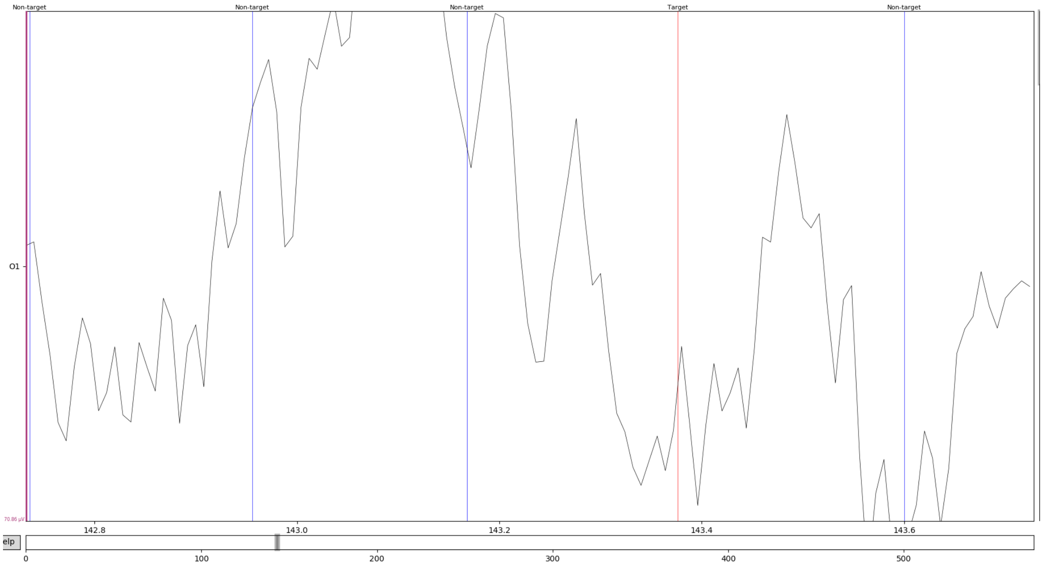

Figure 1: Plotted filtered signal on O1 channel with markers

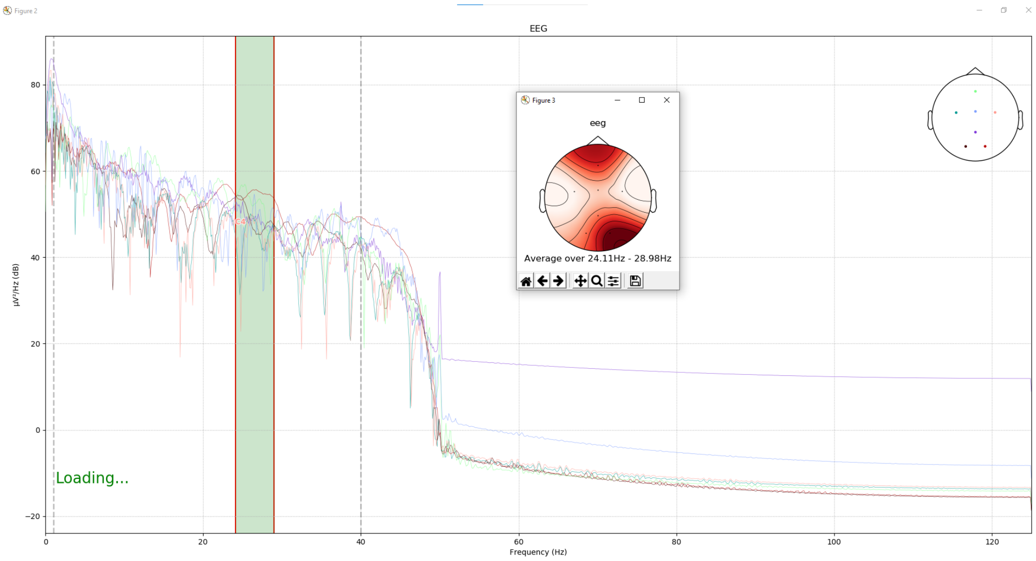

Figure 2: PSD plot with montage