Intro to Neuropype

Emmanuel Olateju

January 8, 2022

Contents

Chapter 1

DataFlow

Neuropype is a dataflow based programming suite for processing of biological signals employing objects called nodes which are interconnected and present in a program called a pipeline to distribute data, alter direction of dataflow and transform data. The pipeline program which contains and stores the nodes and ther connections is created and written through the use of a suite called the Neuropype pipeline designer
Neuropype offers a variety of node types(signal processing, machine-learning, visualization, data-formatting nodes, etc). It also provides methods of developing user custom nodes using the python programming language. All these nodes accept input data and spit out some form of output data.
There is usually some form of input node that reads the input data-file or livestream from which other nodes receive this input. Nodes can accept data either as a whole batch(a large chunk of data) or in bits as small successive chops of data and also output data as a whole batch or as sequences of small data chops at regular time intervals. Nodes can have more than one input with each input accepting different kind of dataflow.
There is no restriction to the form of dataflow within a single pipeline.
So as to use neuropype we have to understand how the data flows between the nodes in the pipeline program. Is data received/sent by a node as a whole or in mini-batches(chunks)? This is the subject of discuss in this chapter.

1.1 streaming data

Flow of data in neuropype can be continuous, in mini-batch or otherwise sending chunks or parts of the data at regular time intervals. This kind of data flow is called streaming dataflow or onlline processing. It is mostly used with real-time input biosignal data.

In every 0.04s(25 times in one 1s), unless the pipeline is paused, each node accepting a streaming input is passed the next successive chop of data and then moves whatever data it outputs into the next processing node.

Some common nodes in neuropype that are designed to handle streaming data are LSL input, FIR filter etc. (Note that most nodes in neuropype can handle streaming and non-streaming data)

1.2 offline data

Dataflow is not continuous, instead the whole large chunk of data is either passed in without being broken into smaller parts or data is output as a whole large chunk of data. This is often called offline processing and is mostly used in training or supervised learning tasks.
The import xdf, import csv are nodes that take in batch data input from files and outputs batch data.
As mentioned earlier, most nodes in Neuropype can accept streaming data or batch(offline) data, some examples are FIR filter, segmentation,Time plot, etc.

1.3 DataFlow Conversions

Some nodes can accept batch data input, store them in memory and output them in regular time intervals as little chops of data from the large chunk of data input.An example of such node is 'Stream data',etc.
Some nodes do the reverse, which is accept streaming data and storing them in memory and then spitting out the whole large chunk of accumulated streaming data, usually with time-stamps. An example of such a node is 'Record to csv'.

Chapter 2

Nodes

We previously discussed the form of data that can be accessed and given out by nodes in neuropype. Much more than that, the nodes can be classified by the functions they perform.
Neuropype pipeline designer groups nodes according to the class of function they perform, some common groups of nodes are feature-extraction, signal-processing, machine-learning, formatting, visualization, etcc.

2.1 processing nodes

These are nodes that transform their data input. These nodes accept packets(successive small chops) of data or a whole recording(large chunk) of data, spitting out the transformed data with dataflow similar to that of the input.
Some examples are FIR filter,squaring,Moving Average,squaare root, mean etc.
Note that a node may not pass data in just one of the dataflow forms discussed in the previous chapter, a node may pass both batch data and streaming data.

2.2 dataflow conversion nodes

As mentioned before these group of nodes alter the form of dataflow from input to their output. Its possible they perform some processing of data while doing so.

2.3 output/visualization nodes

These group of nodes accept any form of dataflow as input to produce some sort of visual representation of their input data. Some examples of such nodes are TimePlot,PrintToConsole etc.

2.4 input nodes

These group of nodes read input biological signals either from some stored recording file or from a livestrem of data from an instrument
An example is import xdf which spits out a whole large chunk of data which is the whole recording from a stored .xdf file.
Another is LSL input which accepts livestream data from an EEG recording device and spits the data out at regular time-intervals.

2.5 marker action based nodes

These nodes are for manipulating event labels in the input data or manipulating data based on the associated event-label. Example nodes are Assign-target, segmentation, Accumulate calibration data, etc

Chapter 3

Data Format in Neuropype

3.1 tensors

2D arrays are mostly multi-channel time series. In some pipelines, we can have 3D arrays and more. General n-way arrays in neuropye are also called tensors

3.2 axes

The tensors are not just grid of raw numbers each axes has a type,i.e there are exes type in neuropype. There exists nodes to map an axis type to another such as the override axis node. Below is a list of neuropype pipeline designer axes types:

- axis[generic axis type]
- time
- space
- frequency
- instance
- channel label
- feature
- lag
- statistic

Chapter 4

Lab Streaming Layer(LSL)

LabStreamingLayer(LSL) is a system for unified collection of time-series data in reseach exeperiments that hndles networking, time-synchronization, (near) real-time access, optionally centralized collection, viewing and recording of the data.

4.1 LabStreamingLayer API

A LSL object has the following attributes

- Sample: A single measurement of all channels from a device. The sample can also be time associated event Markers. These time associated event markers are usually sent over a seperate stream of type marker while multichannel time-series data usually have a stream type of EEG.
- Metadata: XML data containing information about the stream
- Stream: Combination of sampled data from a device with the metadata
- Stream Outlet: Makes time-series data streams available on lab network, pushing data sample by sample into outlet. An outlet makes the stream visible and available to computer son the network. The stream outlet canbe subscribed to using a resolver and connecting a Stream inlet to it.
- Stream Inlet: This receives receives time-series data from a single stream outlet. Asides from the samples, the metadata can be obtained.
- Resolver: This resolves streams present o the lab network according to metadata information of the streams like name, content-type, etc.

4.2 Stream Outlet and Stream Inlet

4.2.1 Stream outlet code in python

Below is a code showing EEG data and marker being sent on two different outlet streams:

```
from pylsl import StreamInfo, StreamOutlet

markers=['rest','left-squeeze','right-squeeze']
markerInfo=StreamInfo(name='eegMarker',type='Markers',channel_count=1
source_id='t8u43t98u')
merkerOutlet=StreamOutlet(markerInfo)

noEEGchannels=16
eegInfo=StreamInfo(name='eegSample',type='EEG',channel_count=noEEGch
source_id='myuidt8u43t98u')
eegOutlet=StreamOutlet(eegInfo)

while True:
    eegSample=[rand() for in range(noEEGchannels)]
    markerSample=[markers[random.choice(range(len(markers)))]]
    eegOutlet.push_sample(eegSample)
    markerOutlet.push_sample(markersample)
```

4.2.2 Stream inlet code in python

Below is a code showing EEG data being received from an inlet stream of name eegSample and type EEG

```
from pylsl import StreamInlet, resolve_stream

streams=resolve_stream('type','EEG')
inlet=StreamInlet(streams[0])
while True:
    # get a new sample (you can also omit the timestamp part if you'r
    # interested in it)
    sample, timestamp = inlet.pull_sample()
    print(timestamp, sample)
```

4.3 Lab Recorder

The Lab recorder is a softtware that comes along with the LSL suite and is used to merge data from various LSL stream outlets and recording them into a single .xdf file. A simple example of this is EEG data from an openBCI EEG on an LSL stream while also using the code in 4.2.1 while omitting the eegOutlet in

4.2.1 to generate a marker stream outlet. These streams are merged together, time-synchronized and then recorded into a single .xdf file.

Chapter 5

Typical Use of Neuropype Pipeline Designer

5.1 LSL inputs and output

Neuropype come with LSL input and output nodes found in the network group of nodes. The LSL input node takes in data from the appropriate LSL strem-outlet if available and then spits out streaming dat to successive nodes. In the LSL input node settings, the name of a marker stream-outlet can be added, which is also comes out of the LSL input node output. The LSL output node takes in streaming data and creates a LSL stream-outlet on the LSL network.

5.2 Offline processing

Offline processing usually starts with an import node, in our case an import XDF node found in the File System group of nodes. The batch data at the output of the import node is passed to processing nodes directly without using any method to break them sequences that can be streamed based on a clock. Below is an example of a pure offline rocessing pipeline.

5.3 Online processing

Online processing usually starts with an LSL input node.It can also start with an import node after which the batch data output from the input node is converted to streaming data using a stream data node found in the formatting group of nodes.

5.4 Accumulate and Inject Calibration Data

The AccumulateCalibrationData and InjectCalibrationData nodes are found in the machine-learning group of nodes. They are used for calibration of processing nodes.

5.4.1 AccumulateCalibrationData

The AccumulateCalibrationData node are always used to calibrate processing nodes when we have streaming data and intend to collect the calibration data live at the beginning of a session instead of using a recording of previously acquired data.
The AccumulateCalibrationData node in its settings is told what markers indicate the beginning of calibration data and what marker indicates the end of calibration data.The node buffers data from the stream into one chunk from the beginning marker to the end and then spits the calibration chunk data to successive nodes for calibration after which the AccumulateCalibrationData node just passes streaming data as streaming data.
Some processing nodes are capable of incremental calibration on streaming data and therfore do not require an AccumulateCalibrationData node for calibration from streaming data.

5.4.2 InjectCalibrationData

The InjectCalibrationData unlike the AccumulateCalibrationData has two inputs. One input takes in calbration data as a batch data(large chunk) using an import node which first passes through to successive nodes that require calibration.
The second input takes in streaming data which is passed through to successive nodes after the calibration data has passed through, then the pipeline does its regular processing.