sEEG_animate_onset_2D

November 19, 2025

[1]:
```
from numpy import pi, linspace, sin, cos, arctan2, degrees, diff, arange, c_,␣
 ↪array, asarray, delete, argmax, around
from numpy import zeros, zeros_like, ones, correlate, std, exp, log,␣
 ↪corrcoef,fill_diagonal, gradient, unravel_index, insert
from numpy import sort, concatenate, angle, unwrap, mod, arctan2, degrees,␣
 ↪sqrt, digitize, roll, percentile, insert, cumsum

from pandas  import read_csv

import json

import matplotlib.pyplot as plt
from   matplotlib.animation import FuncAnimation
from   IPython.display import HTML
```

0.1 sEEG data

[2]:
```
sz = 0

sr = 1000       # samples per second

patient_load = "HS_2018wuzhisen"

folder = '/Users/geroldbaier/Library/CloudStorage/Dropbox/EEG/EEG_Data/' +␣
 ↪patient_load + '/'

path = folder + 'Series/WZS_Seizure_' + str(sz+1) + '.csv'

data_raw = read_csv(path, delimiter=r"\s+")
data_raw = data_raw.values

rows, chans = data_raw.shape

time = linspace(0, rows//sr, rows)

labels_chars =  folder + 'Series/WZS_Labels.txt'
```

```
all_labels_list = []

with open(labels_chars, 'r') as file:

    for line in file:

        label = line[:-1]
        all_labels_list.append(label)

label_dict_chars = folder + 'Series/All_onsets_label_dict.txt'

with open(label_dict_chars, "r") as dict_file:

    labels_dict_read = json.load(dict_file)


folder = '/Users/geroldbaier/Library/CloudStorage/Dropbox/EEG/EEG_Data/' +␣
 ↪patient_load + '/'

onset_chars = folder + 'onset_reduced.txt'

onset_times = read_csv(onset_chars, header=None)

onset_time = onset_times[0][sz]

patient = 'HS_2018WZS' # HS_2018WZS, Tumor_2017LZJ, FCD_2019ZPY

if patient == 'HS_2018WZS':

    faulty_channel = (10, 44, 45)

    data_raw = delete(data_raw, faulty_channel, axis=1)

    chans -= len(faulty_channel)

    all_labels_list = delete(all_labels_list, faulty_channel)

data_raw.shape, len(all_labels_list)
```

[2]:
```
((294037, 67), 67)
```

[3]:
```
def data_band_pass_filter_2d(data_2d, cut_low, cut_high, order, sr):


    from scipy.signal import butter, sosfiltfilt
    from numpy import arange, zeros, asarray
```

```
    sos = butter(order, (cut_low, cut_high), btype='bandpass', fs=sr,␣
 ↪output='sos')

    data_filtered = sosfiltfilt(sos, data_2d, padlen=100, axis=0)

    return data_filtered
```

[4]:
```
freq_bands = ((0.001, 0.5), (0.5, 13))

# Embedding
dims = 2

# Filter order
order = 5

data_VdV = zeros((data_raw.shape[0], data_raw.shape[1], dims, len(freq_bands)))
data     = zeros((data_raw.shape[0], data_raw.shape[1], dims, len(freq_bands)))

for index, band in enumerate(freq_bands):

    data_band = data_band_pass_filter_2d(data_raw, freq_bands[index][0],␣
 ↪freq_bands[index][1], order, sr)

    data_band         = asarray(data_band)
    data_band_grad    = gradient(data_band, axis=0)

    data_VdV[:,:,0,index] = data_band
    data_VdV[:,:,1,index] = data_band_grad

dim       = 0
freq_band = 1
dict_min  = 7

data_VdV_2D = zeros((data_VdV.shape[0], dict_min, len(labels_dict_read)))

for index, key in enumerate(labels_dict_read):

    data_VdV_2D[:, :, index] = data_VdV[:, labels_dict_read[key][0]:
 ↪labels_dict_read[key][0]+dict_min, dim, freq_band]



data_2D = zeros((data_VdV.shape[0], dict_min, len(labels_dict_read)))

for index, key in enumerate(labels_dict_read):
```

```
    data_2D[:, :, index] = data_VdV[:, labels_dict_read[key][0]:
 ↪labels_dict_read[key][0]+dict_min, dim, freq_band]

print(data_2D.shape)
print('Rows, Dim 1, Dim 2')
```

```
(294037, 7, 6)
Rows, Dim 1, Dim 2
```

[ ]:

1 Pseudo 2D heatmap

[5]:
```
time_steps = 1000

win_start = 160000

downsample = 10

rows, cols = data_2D.shape[1:]

data_surface = data_2D[win_start:win_start+time_steps:downsample, :, :].copy()

vmin = 0
vmax = 0.8*data_surface.max()

fig, ax = plt.subplots(figsize=(7, 6))

plt.close()

# Heatmap
im = ax.imshow(data_surface[0], cmap='bwr', aspect='auto',␣
 ↪interpolation='Gaussian', vmin=vmin, vmax=vmax)
ax.set_title(f'Votages, Time step: 0')
fig.colorbar(im, ax=ax, shrink=0.5)

def update(frame):
    im.set_array(data_surface[frame])
    ax.set_title(f'Time step: {frame}')
    return [im]

# Animation
anim = FuncAnimation(
    fig,
    update,
    frames=len(data_surface),
    interval=100,  # msecs between frames
```

```
    blit=True
)

# Display
HTML(anim.to_jshtml())
```

[5]: <IPython.core.display.HTML object>

[ ]: