project_2

October 8, 2024

Instituto Tecnológico de Aeronáutica - ITA

Divisão de Engenharia Eletrônica - IEE

ET-287 - Processamento de sinais usando redes neurais

Professora Sarah Negreiros de Carvalho Leite

Aluno Felipe Keller Baltor

1 Projeto 2

1.0.1 *1. Baixe os sinais cerebrais de um indivíduo disponível em: https://bci.med.tsinghua.edu.cn/download.html.*

[1]:
```
%reset
```

```
Once deleted, variables cannot be recovered. Proceed (y/[n])? y
```

[2]:
```
# Importações básicas

import numpy as np # biblioteca de manipulação vetorial e numérica
import matplotlib.pyplot as plt # biblioteca para traçar gráficos
import pandas as pd # biblioteca de manipulação de dados tabulares
from pathlib import Path # biblioteca para manipulação de "paths"
import urllib3 # biblioteca para download do dataset
from tqdm import tqdm # barra de download
import py7zr # descomprimir 7z
import scipy
```

[3]:
```
# Checando se estamos no diretório correto

project_dir = Path('.')

assert project_dir.resolve().name == 'project_2'
```

[5]:
```
# Baixando os dados

# Definindo o arquivo a ser baixado
subject = 'S1'
data_dir = project_dir / 'data'

if not data_dir.is_dir():
    print("Data file doesn't exists, checking for compressed file")

    data_compressed = project_dir / f'{subject}.mat.7z'
    if not data_compressed.is_file():
        data_dir.mkdir(exist_ok=True)
        print("Compressed file doesn't exists, downloading...")

        http = urllib3.PoolManager()
        CHUNK_SIZE = 2**16

        data_description_url = 'https://bci.med.tsinghua.edu.cn/upload/yijun/
 ↪Readme.txt'
        data_description_file = data_dir / 'readme.txt'
        resp = http.request('GET', data_description_url, preload_content =␣
 ↪False)
        with open(data_description_file, 'wb') as file:
            while True:
                data = resp.read(CHUNK_SIZE)
                if not data:
                    break
                file.write(data)
        resp.release_conn()

        freq_phase_url = 'https://bci.med.tsinghua.edu.cn/upload/yijun/
 ↪Freq_Phase.mat'
        freq_phase_file = data_dir / 'freq_phase.mat'
        resp = http.request('GET', freq_phase_url, preload_content = False)
        with open(freq_phase_file, 'wb') as file:
            while True:
                data = resp.read(CHUNK_SIZE)
                if not data:
                    break
                file.write(data)
        resp.release_conn()

        data_url = f'https://bci.med.tsinghua.edu.cn/upload/yijun/
 ↪{data_compressed}'
        resp = http.request('GET', data_url, preload_content = False)
        TOTAL_SIZE = int(resp.headers.get('Content-Length'))
        with (
```

```
            open(data_compressed, 'wb') as file,
            tqdm(
                total = TOTAL_SIZE,
                desc = f'Downloading {data_compressed.name}',
                unit = 'B',
                unit_scale = True) as bar
        ):
            for chunck in resp.stream(CHUNK_SIZE):
                size = file.write(chunck)
                bar.update(size)
        resp.release_conn()

    print("Decompressing file...")
    with py7zr.SevenZipFile(data_compressed.name, 'r') as compressed:
        compressed.extract(path = data_dir, recursive = False)

    print("Data file ready!")

else:
    print('Data already exists!')
```

```
Data already exists!
```

1.0.2 *2. Organize sua matriz de dados de entrada x e seu valor de rótulos y da seguinte maneira...*

[6]:
```
# Descrição dos dados
# Repare que não foi aplicada nenhum 'encoding' posto que o arquivo parece
# ter uma codificação desconhecida. Esquemas tradicionais como utf-8 e␣
 ↪iso-8859-1
# não foram capazes de exibir corretamente os caracteres.

data_description_file = data_dir / 'readme.txt'
with open(data_description_file, 'rb') as f:
    descr = f.read()
    print(descr)
```

```
b'This dataset gathered SSVEP-BCI recordings of 35 healthy subjects (17 females,
aged 17-34 years, mean age: 22 years) focusing on 40 characters flickering at
different frequencies (8-15.8 Hz with an interval of 0.2 Hz). For each subject,
the experiment consisted of 6 blocks. Each block contained 40 trials
corresponding to all 40 characters indicated in a random order. Each trial
started with a visual cue (a red square) indicating a target stimulus. The cue
appeared for 0.5 s on the screen. Subjects were asked to shift their gaze to the
target as soon as possible within the cue duration. Following the cue offset,
all stimuli started to flicker on the screen concurrently and lasted 5 s. After
stimulus offset, the screen was blank for 0.5 s before the next trial began,
```

which allowed the subjects to have short breaks between consecutive trials. Each trial lasted a total of 6 s. To facilitate visual fixation, a red triangle appeared below the flickering target during the stimulation period. In each block, subjects were asked to avoid eye blinks during the stimulation period. To avoid visual fatigue, there was a rest for several minutes between two consecutive blocks.\r\n\r\nEEG data were acquired using a Synamps2 system (Neuroscan, Inc.) with a sampling rate of 1000 Hz. The amplifier frequency passband ranged from 0.15 Hz to 200 Hz. Sixty-four channels covered the whole scalp of the subject and were aligned according to the international 10-20 system. The ground was placed on midway between Fz and FPz. The reference was located on the vertex. Electrode impedances were kept below 10 K\xa6\xb8. To remove the common power-line noise, a notch filter at 50 Hz was applied in data recording. Event triggers generated by the computer to the amplifier and recorded on an event channel synchronized to the EEG data. \r\n\r\nThe continuous EEG data was segmented into 6 s epochs (500 ms pre-stimulus, 5.5 s post-stimulus onset). The epochs were subsequently downsampled to 250 Hz. Thus each trial consisted of 1500 time points. Finally, these data were stored as double-precision floating-point values in MATLAB and were named as subject indices (i.e., S01.mat, \xa1\xad, S35.mat). For each file, the data loaded in MATLAB generate a 4-D matrix named \xa1\xaedata\xa1\xaf with dimensions of [64, 1500, 40, 6]. The four dimensions indicate \xa1\xaeElectrode index\xa1\xaf, \xa1\xaeTime points\xa1\xaf, \xa1\xaeTarget index\xa1\xaf, and \xa1\xaeBlock index\xa1\xaf. The electrode positions were saved in a \xa1\xae64-channels.loc\xa1\xaf file. Six trials were available for each SSVEP frequency. Frequency and phase values for the 40 target indices were saved in a \xa1\xaeFreq_Phase.mat\xa1\xaf file.\r\n\r\nInformation for all subjects was listed in a \xa1\xaeSub_info.txt\xa1\xaf file. For each subject, there are five factors including \xa1\xaeSubject Index\xa1\xaf, \xa1\xaeGender\xa1\xaf, \xa1\xaeAge\xa1\xaf, \xa1\xaeHandedness\xa1\xaf, and \xa1\xaeGroup\xa1\xaf. Subjects were divided into an \xa1\xaeexperienced\xa1\xaf group (eight subjects, S01-S08) and a \xa1\xaenaive\xa1\xaf group (27 subjects, S09-S35) according to their experience in SSVEP-based BCIs.\r\n'

[7]:
```
# Dados do mapeamento de frequência e fase

freq_phase_file = data_dir / 'freq_phase.mat'
freq_phase = scipy.io.loadmat(freq_phase_file)

print(freq_phase['phases'].shape)
print(freq_phase['freqs'].shape)
print(freq_phase['freqs'])
```

```
(1, 40)
(1, 40)
[[ 8.   9.  10.  11.  12.  13.  14.  15.   8.2  9.2 10.2 11.2 12.2 13.2
  14.2 15.2  8.4  9.4 10.4 11.4 12.4 13.4 14.4 15.4  8.6  9.6 10.6 11.6
  12.6 13.6 14.6 15.6  8.8  9.8 10.8 11.8 12.8 13.8 14.8 15.8]]
```

[8]:
```
# Dados experimentais

subject_file = data_dir / f'{subject}.mat'
subject_mat = scipy.io.loadmat(subject_file)
subject_mat['data'].shape
```

[8]:
```
(64, 1500, 40, 6)
```

[9]:
```
# Array de entrada 'x'

freq1 = 0
freq2 = 1

data1 = subject_mat['data'][60, 125:1375, freq1]
data1 = np.swapaxes(data1, 0, 1)

data2 = subject_mat['data'][60, 125:1375, freq2]
data2 = np.swapaxes(data2, 0, 1)

x = np.vstack((data1, data2))

x.shape, x, x.max(), x.min(), x.mean(), x.std()
```

[9]:
```
((12, 1250),
 array([[  3.70141888,   0.50322926,  -0.27934119, …,   1.9592433 ,
           1.01149976,  -5.88010216],
        [ -6.64295244,  -2.83018279,   0.04418454, …,   6.31171703,
           3.12776113,   2.56537223],
        [  8.17666054,   4.36537313,   8.2611742 , …,  -8.82727051,
          -7.03082752,  -8.43020248],
        …,
        [ -7.64600086,  -1.01372385,  -2.60362577, …,  10.36601925,
           6.28191471,   4.59104061],
        [ 15.23135948,  16.80391312,  24.1235714 , …,  10.85264206,
           7.32025146,   9.7661171 ],
        [-20.64233398, -15.26442337, -18.02239799, …,   6.99568748,
          -2.47388935,  -6.33217812]]),
 33.8286018371582,
 -48.90739059448242,
 -0.39478745126182524,
 9.030378711934027)
```

[10]:
```
# Array de rótulos 'y'

y = np.array([1 if n < 6 else -1 for n in range(12)])

y.shape, y
```

[10]:
```
((12,), array([ 1,  1,  1,  1,  1,  1, -1, -1, -1, -1, -1, -1]))
```

1.0.3 *3. Implemente o perceptron que receba em sua entrada a matriz de dados x e retorne o valor de y, indicando a frequência de estimulação f_1 ou f_2.*

1.0.4 *Separe aleatoriamente 4 amostras de cada classe para treinar o perceptron e o restante das amostras para validar o sistema.*

[11]:
```
class Perceptron:
    def __init__(self, input_dim, learning_rate, activation_function = None):
        self.input_dim = input_dim
        self.learning_rate = learning_rate
        self.w = np.ones(input_dim + 1)
        if activation_function is not None:
            self.activation_function = activation_function
        else:
            self.activation_function = lambda x: -1 if x < 0 else 1

    def train(self, inputs, outputs):
        assert len(inputs) == len(outputs)

        for i in range(len(inputs)):
            x = inputs[i]
            y = outputs[i]

            x = np.append(x, 1)
            u = np.dot(self.w, x)
            y_pred = self.activation_function(u)
            error = y - y_pred
            update = self.learning_rate * error * x
            self.w += update

    def predict(self, inpt):
        inpt = np.append(inpt, 1)
        u = np.dot(self.w, inpt)
        return self.activation_function(u)
```

[332]:
```
rng = np.random.default_rng()

x_train_index_freq1 = rng.choice(range(6), size = 4, replace = False)
x_train_freq1 = x[x_train_index_freq1]

x_train_index_freq2 = rng.choice(range(6, 12), size = 4, replace = False)
x_train_freq2 = x[x_train_index_freq2]

x_train = np.vstack((x_train_freq1, x_train_freq2))
y_train = np.array([1 if n < 4 else -1 for n in range(8)])
```

```
p = Perceptron(1250, learning_rate = 0.5)
p.train(x_train, y_train)

x_test_index = np.setdiff1d(
    np.array(range(0, 12)),
    np.concatenate((x_train_index_freq1, x_train_index_freq2))
)

x_test = x[x_test_index]
y_test = np.array([1 if n < len(x_test)/2 else -1 for n in range(len(x_test))])

for i in range(len(x_test)):
    print(y_test[i], p.predict(x_test[i]))
```

```
1 1
1 1
-1 -1
-1 -1
```

[ ]: