The Ninapro Dataset

Can be found here

In this week we used the Ninapro dataset DB1.

```
import os

root_dir = os.getcwd()
rel_dir = 'datasets/ninapro/db1/s1/S1_A1_E1.mat'
file_path = os.path.join(root_dir, rel_dir)

import numpy as np
from scipy.io import loadmat as ld

mat_data = ld(rel_dir, appendmat=False)
```

Available fields in the ninapro dataset

```
keys = list(mat_data.keys())
keys
```

```
['__header__',
 '__version__',
 '__globals__',
 'emg',
 'stimulus',
 'glove',
 'subject',
 'exercise',
 'repetition',
 'restimulus',
 'rerepetition']
```

Loading all the subjects' data into a 'struct'.

This struct 'data', will have the following structure:

Data is an array of n subjects each subject containing 3 experiments each experiment containing a dictionary (key-value pair) we'll load the sEMG signals from all the channels, and the cyberglove data, for the hand joint angle regression task.

We also computed the angle differences between samples, to model the correlation between joint angle variation and the incoming EMG signal. This variation per sample is given by the following equation:

\begin{align*} \delta_i\left[n\right] &= \begin{cases} 0 &\quad,\text{if } n \leq 1 \\ g_i\left[n+1\right]-g_i\left[n\right] \end{cases} \end{align*}

Where i is a cyberglove joint ($i \in [1,22), i \in R$), $\delta[n)$ is the angle variation (in $\degree$) at sample n, $g[n)$ is the cyberglove's uncalibrated angle at sample n.

Normalize data

The emg signal was normalized from each channel's relative minimum and max, to $[0,1)$ while the glove angles were normalized, from $[0,360)$ to $[0,1)$, using:

$$x' = \frac{x - \min x}{\max x - \min x}$$

And 'delta', was normalized from $[-360,360)$ to $[-1,1)$, using:

$$x'' = 2\frac{x - \min x}{\max x - \min x}$$

(source: https://stats.stackexchange.com/questions/178626/how-to-normalize-data-between-1-and-1)

```
import os
from scipy.io import loadmat
import numpy as np

# Define the structure array to store the data
num_subjects = 27
data = []

def normalizer(data_array,min_val,max_val,type):

    match type:
        case "uni":
            return (data_array - min_val) / (max_val - min_val)

        case "bi":
            return 2 * ( (data_array - min_val) / (max_val - min_val )
) - 1
        case __:
            raise Exception(f"{type} type normalization not yet
implemented.")

# Loop through each subject
for subj_idx in range(1, num_subjects + 1):
    # Set up the subject directory name and path
    subject_dir = f"s{subj_idx}"
    subject_path = os.path.join("datasets", "ninapro", "db1",
subject_dir)

    # Initialize lists to hold the experimental data
    exps = []
```

```
    # Loop through each experiment for the current subject
    for exp_idx in range(1, 4):
        # Set up the experiment file name and path
        exp_name = f"S{subj_idx}_A1_E{exp_idx}.mat"
        exp_path = os.path.join(subject_path, exp_name)

        # Load the MATLAB file
        mat_file = loadmat(exp_path)

        # Extract the relevant information from the loaded dictionary
        emg = mat_file['emg']

        glove = mat_file['glove']

        # glove delta
        delta = np.diff(glove, axis=0) # compute differences
        delta = np.pad(delta, ((1, 0), (0, 0)), mode='constant') #
first sample delta = 0

        # normalize data
        norm_emg = normalizer(emg, np.min(emg, axis = 0), np.max(emg,
axis = 0) , "uni") # min-max channel, [0;1]
        #norm_glove = normalizer(glove,0,360, "uni") # min-max
channel, [0;1]
        norm_delta = normalizer(delta,-360,360, "bi")

        # Add the extracted data to the list of experiments
        exps.append({
            'emg': norm_emg,
            'glove_delta': glove
        })

    # Append the list of experiments to the data list
    data.append(exps)

# Convert the data list to NumPy arrays
data = np.array(data)
```

Next steps

We now have the EMG signals to use as input to the network, and the dependent variable for regression, the $\delta[n]$. This should suffice for the finger regression task, however, we'll try different approaches, such as:

- Different network architectures: Vanilla RNN, LSTM, Bi-directional LSTM
- Use of covariates, such as the physiological data of the subject (sex, age, height, weight, available in the ninapro dataset DB1), to try to improve inter-subject accuracy.
- Use of amputated patient data from the Ninapro dataset.

The next step is to break down the data into batches, and then try training a RNN with differnt input sequence lenghts.