Příloha 9 - Kódové provedení kalibrace

August 11, 2022

1 Basic Imports

[ ]:
```
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings("ignore")

from STUDNA.detection_tools import ratio_method, original_method
from STUDNA.calibration import calibrate_threshold, apply_calibration,␣
↪calibrate_whole_system
```

2 Data importation

[ ]:
```
data = pd.read_csv('measured_data_optimized.csv')
#data['category'] = data['category'].apply(lambda x: x[23:])

# Some editations because of measurement error
data.loc[data['category']=='tighten_after_loose', 'category'] ='orig'
data.loc[data['category']=='two_loosen', 'category'] ='arm'
data.loc[data['category'].apply(lambda x: 'side' in x), 'category'] = 'side'
data.loc[data['category'].apply(lambda x: 'arm' in x), 'category'] = 'arm'


# Code to remove all the measurements from date 18.05.
filt1 = (data['time'].apply(lambda x: x[8:10]) == str(18))
filt2 = data['category'] == 'orig'
filt = filt1 & filt2
data.drop(index=data[filt].index, axis=0, inplace=True)
data.reset_index(drop=True, inplace=True)
data.drop(columns=["time"], inplace=True)
####################


data['category'] = pd.Categorical(data.category)
data.head(3)
```

[ ]:
```
   bench_peaks  actual_peaks  bench_peak_power  actual_peak_power  \
0          0.0         747.0               NaN           0.000038
1          1.0         711.0          0.000033           0.000037
2          2.0         803.0          0.000038           0.000038

   bench_entropy  actual_entropy  bench_w0_mean  actual_w0_mean  bench_w0_std  \
0   6.535392e-07        0.000009       0.001455        0.001479      0.010956
1   7.115839e-07        0.000009       0.001455        0.002000      0.010956
2   8.628529e-07        0.000009       0.001455        0.001687      0.010956

   actual_w0_std  …  w0_frq_0  w0_frq_1  w1_psd_0  w1_psd_1  w1_frq_0  \
0       0.010433  …    2471.6    2475.6  0.066750  0.061347    2482.6
1       0.009401  …    2478.6    2465.7  0.061367  0.057666    2478.6
2       0.010293  …    2473.6    2475.6  0.094203  0.058388    2463.7

   w1_frq_1  nfrq  nw  noise_std  category
0    2481.6     2   2   0.008818       arm
1    2462.7     2   2   0.008819       arm
2    2460.7     2   2   0.008818       arm

[3 rows x 62 columns]
```

[ ]:
```
# Only for optimized
data = original_method(data, 2, 2)
data.head(3)
```

[ ]:
```
   peaks  peak_power   w0_mean    w0_std   w1_mean    w1_std   w0_rstd  \
0  747.0    0.000038  0.001479  0.010433  0.855424  0.000298  0.005095
1  711.0    0.000037  0.002000  0.009401  0.856441  0.000636  0.005319
2  803.0    0.000038  0.001687  0.010293  0.856005  0.000212  0.005213

      w0_50     w0_Ra   w0_Skew  …  w0_frq_0  w0_psd_0  w0_frq_1  w0_psd_1  \
0  0.001081  0.004064 -0.009642  …    2471.6  7.312914    2475.6  5.063459
1  0.001772  0.004224 -0.011133  …    2478.6  8.359195    2465.7  6.579518
2  0.001979  0.004144 -0.028920  …    2473.6  6.260987    2475.6  6.092079

   w1_frq_0  w1_psd_0  w1_frq_1  w1_psd_1  category  category_name
0    2482.6  0.066750    2481.6  0.061347         0            arm
1    2478.6  0.061367    2462.7  0.057666         0            arm
2    2463.7  0.094203    2460.7  0.058388         0            arm

[3 rows x 26 columns]
```

3 Calibration testing

Isolate columns by which the detection will be based on.

[ ]:
```
selected_columns = data.columns[5:7].append(data.columns[8:12]).append(data.
↪columns[13:15])
selected_columns
```

[ ]:
```
Index(['w0_rstd', 'w0_50', 'w0_Skew', 'w0_Kurt', 'w1_rstd', 'w1_50', 'w1_Skew',
       'w1_Kurt'],
      dtype='object')
```

[ ]:
```
log = {
    'score': np.zeros(100),
    'precision': np.zeros(100),
    'false_positive': np.zeros(100),
    'false_negative': np.zeros(100)
}

for epoch in range(100):
    results = calibrate_whole_system(data, list(selected_columns), 0.6,␣
↪norm_signal_cat='orig', rareness=4)
    log['precision'][epoch] = results['precision']
    log['false_positive'][epoch] = results['false_positive']
    log['false_negative'][epoch] = results['false_negative']


print('Precision on error/non-error')
print('mean:', np.mean(log['precision']))
print('std:', np.std(log['precision']))
print()

print('False positives:')
print('mean:', np.mean(log['false_positive']))
print('std:', np.std(log['false_positive']))
print()

print('False negatives:')
print('mean:', np.mean(log['false_negative']))
print('std:', np.std(log['false_negative']))
```

```
Precision on error/non-error
mean: 0.8064516129032261
std: 3.3306690738754696e-16

False positives:
mean: 0.19354838709677427
std: 8.326672684688674e-17

False negatives:
mean: 0.0
```

```
std: 0.0
```