Data Analysis-example

February 3, 2019

1 Data Analysis

1.1 Introduction

This notebook is serving as a presentation of GenAlFeaturesSelector's performance.

The aim of this study is analyzing signal features in order to find the ones that maximizes the brain-computer interface classyfing module. TO achieve that, genetic algorithm is used.

It's workflow can be described as:

```
1.Create a population pop.
2.For each individual randomly generate n features.
3.For each individual create a dataset of extracted features.
4.For each individual train neural network.
5.Test neural network using cross-validation.
6.Choose the best individuals.
7.Pair best ones and create new population.
8.Randomly mutate population.
9.Go back to point 3. until satysfying result is founded or maximum number of
iterations is reached.
```

Data consists of 4 datasets, each containing signal from different subject.

```
In [1]: #Let's start with importing necessary modules.
        from BakSys.BakSys import BakardjianSystem as BakSys
        from feat_gen_algorithm import GenAlFeaturesSelector
        from feat_extraction.dataset_manipulation import *
        from feat_extraction.features_extractor import Chromosome
        from analysis_tools import *
        from sklearn.preprocessing import *
        from sklearn.neural_network import MLPClassifier
        from sklearn.pipeline import make_pipeline
        import matplotlib.pyplot as plt
        import numpy as np
        import pandas as pd
        import scipy.stats
        import math
        import time
```

```
In [2]: #Loading raw data
        subj1_raw = load_dataset('datasetSUBJ1.npy')
        subj2_raw = load_dataset('datasetSUBJ2.npy')
        subj3_raw = load_dataset('datasetSUBJ3.npy')
        subj4_raw = load_dataset('datasetSUBJ4.npy')
```

```
In [3]: #Setting parameters

        n_features = 1
        n_population = 7
        desired_fitness = 0.7,
        max_generation = 50
        clf = MLPClassifier(max_iter=800,random_state=42,tol=1e-3)

        #Other parameters I left default, like scaler,classifier,mut_prob

        gafeat = GenAlFeaturesSelector(n_feat=n_features,n_pop=n_population,max_gen=max_generat
                                       desired_fit=desired_fitness,clf=clf)
```

2 Fitting data to model

After preparing data, it's time to fit it to model and obtain results.
Firstly, I will do it for 2 second window.

```
In [4]: time_window = 2

        bs = BakSys(threeclass=False,seconds = time_window)

        subj1 = preprocessing(subj1_raw,'Subject 1 data',256,bs,time_window = time_window)
        subj2 = preprocessing(subj2_raw,'Subject 2 data',256,bs,time_window = time_window)
        subj3 = preprocessing(subj3_raw,'Subject 3 data',256,bs,time_window = time_window)
        subj4 = preprocessing(subj4_raw,'Subject 4 data',256,bs,time_window = time_window)
        overall = (np.vstack([subj1[0],subj2[0],subj3[0],subj4[0]]),
                   np.hstack([subj1[1],subj2[1],subj3[1],subj4[1]]),
                   subj1[2],'Overall data')
```

```
C:\Users\stakar\Anaconda3\lib\site-packages\scipy\signal\_arraytools.py:45: FutureWarning: Usi
  b = a[a_slice]
```

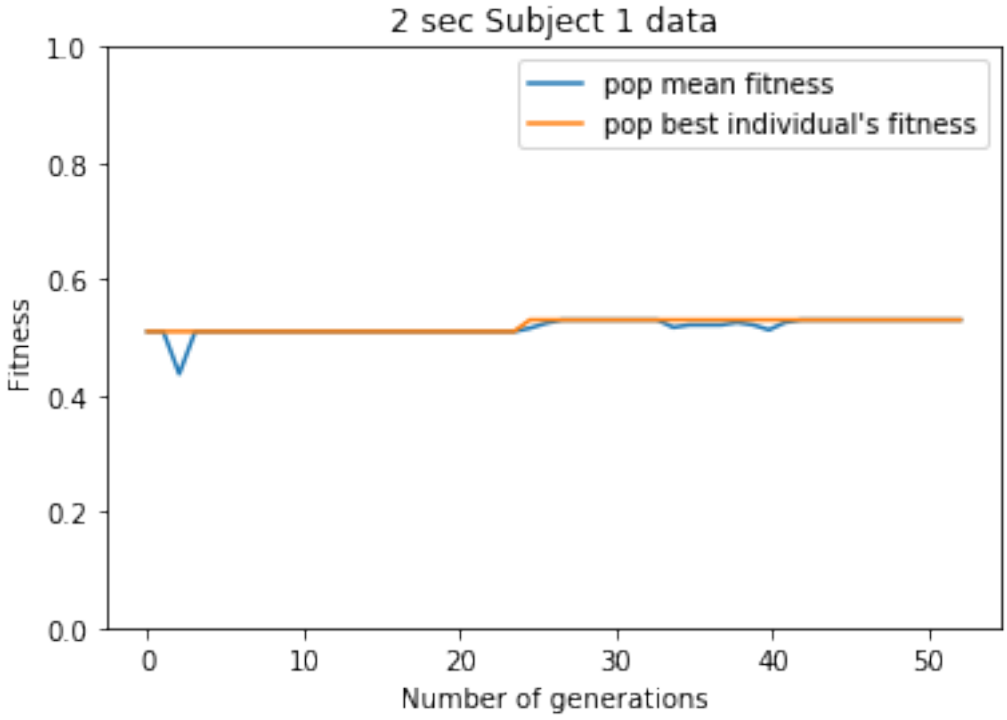
3 Transforming data

For each subject I will transform it, using gafeat class, then plot performance, extract best features, achieved accuracy, number of generations that was taken to find it, and using itr function I will extract how good classifier module perform using chosen features.

```
In [5]: results_2_sec = list()
        best_feat_list_2_sec = list()
```

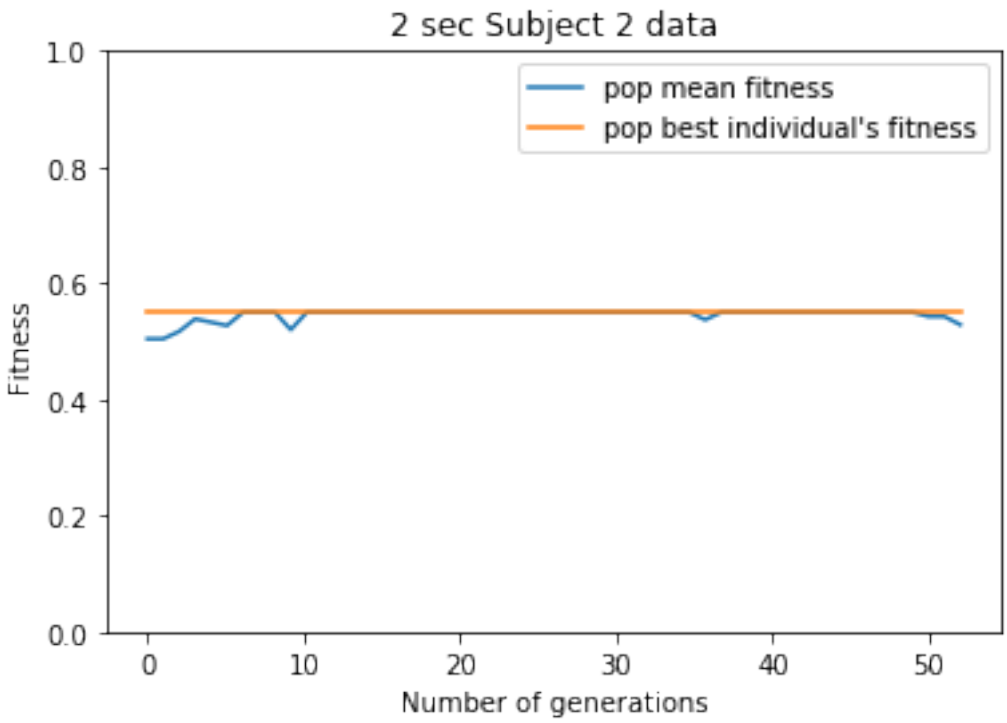
```
        for subject in [subj1,subj2]:
            gafeat.fit_transform(subject[0],subject[1])
            gafeat.plot_fitness('2 sec ' + subject[3])
            plt.show()
            best_feat = pd.Series(best_features(gafeat.pop,gafeat.pop_fit))
            ind = gafeat.pop[np.argmax(gafeat.pop_fit)]
            inf_ratio = itr(bs,subject[2],subject[0],subject[1],ind,clf=clf,
                            accuracy = gafeat.best_ind,time_window=time_window)
            subject_results = {'Score':gafeat.best_ind,
                               'Num of Generations':gafeat.n_generation,
                               'Best features':best_feat.str.cat(sep=',')}
            subject_results['ITR'] = round(inf_ratio)
            results_2_sec.append(subject_results)
            best_feat_list_2_sec.append(best_feat)
```

```
[0.51 0.51 0.51 0.   0.51 0.51 0.51]
[0.53 0.53 0.53 0.53 0.53 0.53 0.53]
```




```
C:\Users\stakar\Anaconda3\lib\site-packages\scipy\signal\_arraytools.py:45: FutureWarning: Usi
  b = a[a_slice]
```

```
[0.51 0.55 0.48 0.51 0.55 0.51 0.51]
[0.55 0.55 0.55 0.55 0.5  0.55 0.55]
```




```
C:\Users\stakar\Anaconda3\lib\site-packages\scipy\signal\_arraytools.py:45: FutureWarning: Usi
  b = a[a_slice]
```

4 Summary

Below I print summary of the algorithm performance.

```
In [9]: summary = pd.DataFrame(columns=list(results_2_sec[0].keys()),
                               index=['Subject1','Subject2'],
                               data=results_2_sec)
        summary
```

```
Out[9]:           Score  Num of Generations     Best features   ITR
        Subject1   0.53                  51      lat,lar,taar   4.0
        Subject2   0.55                  51  lat,lar,par,taar  13.0
```

```
In [ ]:
```