This homework's main work is that it not only gives out the total steps, but also gives out the walk step and running step respectively by first clustering and classifying and applying different filter to running and walking. I tried two filter ways (detrend and Butterworth filter) and three peaks detection algorithm.

(A)Clustering and Classification

a) Aim: Using K-MEANS clustering and the model trained by GNB in Q1 to see when is the person running and when is walking. Thus, I can apply different filter to walking and running respectively.

b)K-Means Code:

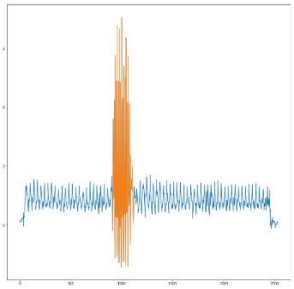
```
#K-MEAN Clustering
    WnR=sliding_win2(load_data(step_data),frame_size,step_size)
    #K-MEAN
    k=2
    kmeans = cluster.KMeans(n_clusters=k)
    kmeans.fit(WnR)
    labels = kmeans.labels_
    centroids = kmeans.cluster_centers_
```

c) GNB classifier code:

```
# Classification about which is running and which is walking
    ds0 =list(np.where(labels==0)[0])
    ds1 =list(np.where(labels==1)[0])
    ds000 = scale(70, ds0)
    ds111 = scale(70, ds1)
    new_step_data_0=step_data.iloc[ds000,:]
    new_step_data_1=step_data.iloc[ds111,:]
    gnb=joblib.load('gnb.model')

    if len(new_step_data_0)> len(new_step_data_1):
        test=new_step_data_0.iloc[:200,1:4]
        WnR_=sliding_win(load_data2(test),200,100)
        result=gnb.predict(WnR_)
        if mean(result)>=3:
            walk_is_0=True
            print('0longer,0 is walk',result)
        else:
            walk_is_0=False
            print('0longer;1 is walk',result)
    else:
        test=new_step_data_1.iloc[:200,1:4]
        WnR_=sliding_win(load_data2(test),200,100)
        result=gnb.predict(WnR_)
        if mean(result)>=3:
            walk_is_0=False
            print('1longer;1 is walk',result)
        else:
            walk_is_0=True
            print('1longer;0 is walk',result)
```

d) Results:



(C) Filtering

Detrend

Butterworth low-pass filter:

For running:

The highest frequency of people running is below 5HZ; Wn=2*fc/fs= 2*5/30=0.3333

For walking:

Wn=0.2

```
77 def BUTTER_LPASS(data,amplingrate=30,label=False):
78
79     fdata=signal.detrend(data)
80     if label== True:
81         WN=0.2
82     if label==False:
83         WN=0.3
84     b, a = scipy.signal.butter(10, WN, 'low')
85     output_signal = scipy.signal.filtfilt(b, a, fdata)
86     return output_signal
```

(D)Find Peaks:

a)Arglmax/argrelmin

The result is not good enough

```
181 #         print(names_[i]+"arg_walk steps:",len(argrelmax(fwalk)[0]))
182 #         print(names_[i]+"arg_run_steps:",len(argrelmax(frun)[0]))
```

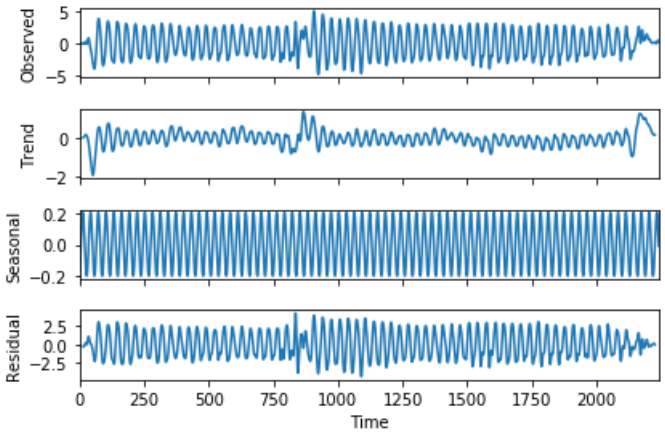
b)Scipy.signal.find_peaks

```
194         # scipy
195
196         walk_step=len(scipy.signal.find_peaks(fwalk, height=0.2, distance=0.25*samplingrate, rel_height=0.5)[0])
197         run_step=len(scipy.signal.find_peaks(frun, height=1,  distance=0.2*samplingrate,rel_height=0.5)[0])
198         print(names_[i]+"scipy_total steps", walk_step+run_step,'  ',"scipy_walk steps:",walk_step,'  ',"scipy_run_steps:",run_step)
199         total_step_.append(walk_step+run_step)
200         walk_step_.append(walk_step)
201         run_step_.append(run_step)
```

c) Signal decomposition

Signal decomposition seems to be more stable. But I am not sure what is its frequency.

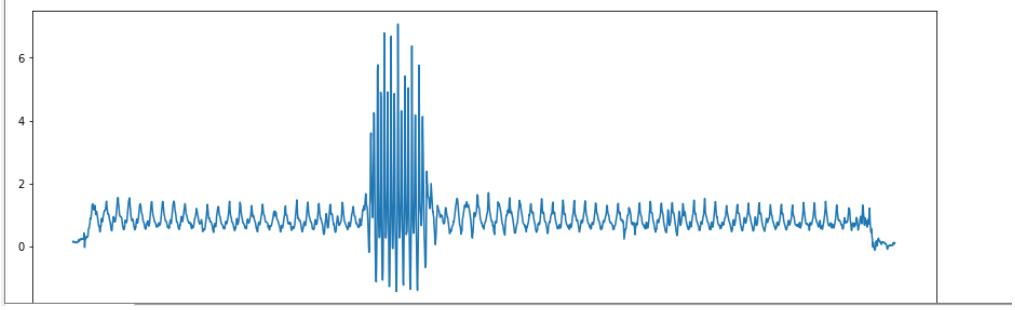
```
183         #Seansonal decomposition
184         result=seasonal_decompose(np.array(fwalk),model='additive',freq=30)
185         season_max=np.max(result.seasonal)
186         season_walk_step=len(np.where(result.seasonal==season_max)[0])
187         result=seasonal_decompose(np.array(frun),model='additive',freq=30)
188         season_max=np.max(result.seasonal)
189         season_run_step=len(np.where(result.seasonal==season_max)[0])
190         print(names_[i]+"season_total_steps", season_run_step+season_walk_step,'  ',"season_walk steps:",season_walk_step,'  ',"season_run_steps:",season_run_step)
191         total_step_.append(season_run_step+season_walk_step)
192         walk_step_.append(season_walk_step)
193         run_step_.append(season_run_step)
```



E)results:

professor's steps

```
acxseason_total_steps 80    season_walk steps: 75    season_run_steps: 5
acxscipy_total steps 78    scipy_walk steps: 65    scipy_run_steps: 13
acyseason_total_steps 79    season_walk steps: 74    season_run_steps: 5
acyscipy_total steps 65    scipy_walk steps: 57    scipy_run_steps: 8
aczseason_total_steps 79    season_walk steps: 74    season_run_steps: 5
aczscipy_total steps 40    scipy_walk steps: 33    scipy_run_steps: 7
gyxseason_total_steps 79    season_walk steps: 74    season_run_steps: 5
gyxscipy_total steps 144    scipy_walk steps: 137    scipy_run_steps: 7
gyyseason_total_steps 80    season_walk steps: 75    season_run_steps: 5
gyyscipy_total steps 124    scipy_walk steps: 119    scipy_run_steps: 5
gyzseason_total_steps 80    season_walk steps: 75    season_run_steps: 5
gyzscipy_total steps 76    scipy_walk steps: 70    scipy_run_steps: 6
AA_scipt_total steps: 45
AG_scipt_total steps: 136
final_result:
 total_step: 83.66666666666667
 walk_step: 77.33333333333333 run_step: 6.333333333333333
```

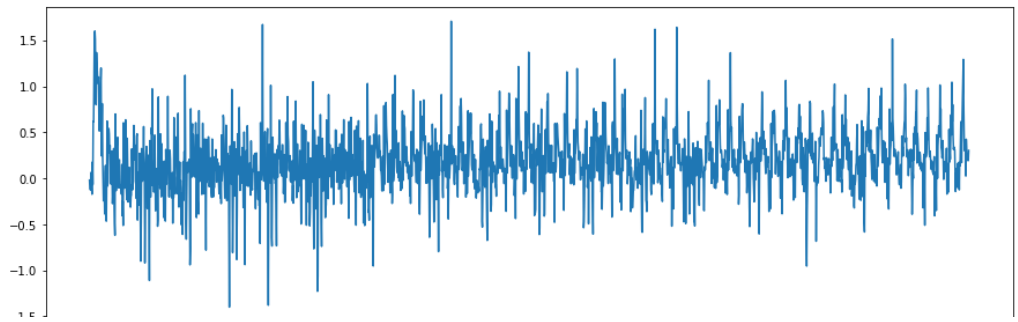


Accuracy:

total step: 83.1% , walk step: 86.1%; run step: 66.7%

My steps

```
acxseason_total_steps 53    season_walk steps: 11    season_run_steps: 42
acxscipy_total steps 16    scipy_walk steps: 16    scipy_run_steps: 0
acyseason_total_steps 55    season_walk steps: 12    season_run_steps: 43
acyscipy_total steps 27    scipy_walk steps: 27    scipy_run_steps: 0
aczseason_total_steps 53    season_walk steps: 11    season_run_steps: 42
aczscipy_total steps 13    scipy_walk steps: 13    scipy_run_steps: 0
gyxseason_total_steps 54    season_walk steps: 12    season_run_steps: 42
gyxscipy_total steps 64    scipy_walk steps: 22    scipy_run_steps: 42
gyyseason_total_steps 54    season_walk steps: 12    season_run_steps: 42
gyyscipy_total steps 94    scipy_walk steps: 31    scipy_run_steps: 63
gyzseason_total_steps 54    season_walk steps: 11    season_run_steps: 43
gyzscipy_total steps 65    scipy_walk steps: 18    scipy_run_steps: 47
AA_scipt_total steps: 147
AG_scipt_total steps: 141
final_result:
 total_step: 50.166666666666664
 walk_step: 16.333333333333332 run_step: 33.833333333333336
```



Accuracy: I can't tell the ground truth.

The algorithm is not accuracy enough at present.

(F) Future work

Try Dynamic Time wrap