ICT for Health
Laboratory # 4
Detect activities

Monica Visintin

Politecnico di Torino

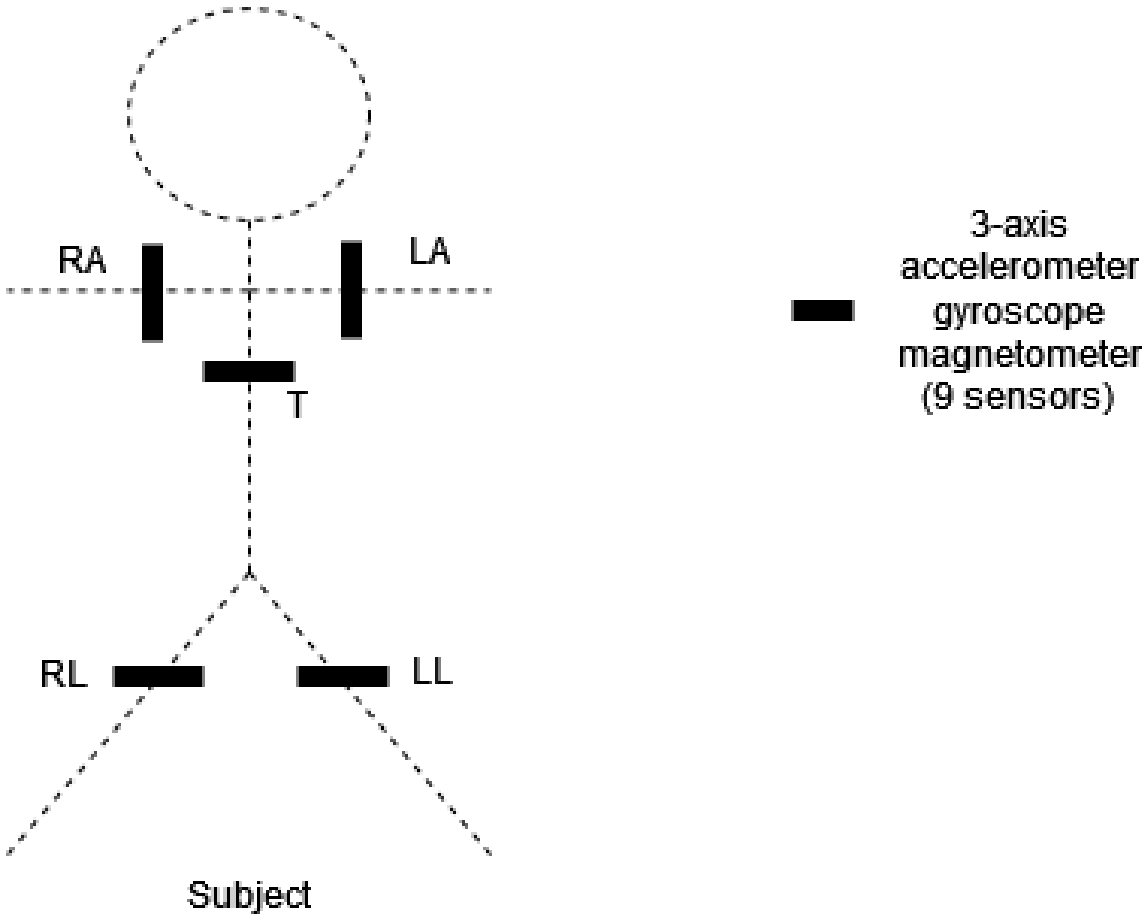
December 9th 2021

Table of Contents

Description [1]

In fitness, people would like to automatically detect their current activity, in order to have a more precise evaluation of the calories, etc.
Task of this lab is to detect the activity (one among 19) starting from the samples taken at frequency 25 Hz from 3-axes accelerometers, gyroscopes, magnetometers placed on torso, right arm, left arm, right leg, left leg: 5 positions, 3 sensors per position, each measuring 3 values on the x-y-z coordinates, total of 45 values for each sample (`https://archive.ics.uci.edu/ml/datasets/Daily+and+Sports+Activities`).
Data were taken from 8 subjects; each subject practised the activity for 5 minutes and slices/files of 5 seconds were created (each slice/file contains 125 rows and 45 columns).

Description [2]



Description [3]

We want a simple algorithm (not neural networks, taught in the parallel class), valid for a specific subject. The idea is to use the clustering algorithm **K-Means**:

- Read the first M slices for the 19 activities of the chosen subject.
- Apply K-Means with 19 clusters and find the 19 centroids.
- For each row of the subsequent slices detect the activity as the one with centroid at **minimum distance** from the row.

This lab is used just as a **feasibility study**. In the real life, an Android/IOs application should be written, that runs on the subject smartphone and gets the data from sensors and makes decisions. The application should have a first training phase in which the subject stores the data, labels them with the activity and applies K-Means. Note that we use a **clustering algorithm** to solve a **classification problem**.

Description [4]

Less than 15 sensors/45 features can be used (the fewer, the better).
Data can/should be **processed** before application of K-Means (for example, through smoothing, enhancing peaks, reshaping, etc) .
You cannot use machine learning algorithms other than K-Means, because simplicity is essential (less energy to perform the computations, batteries last longer, less required CPU, etc).
You can make a decision at each timestamp (25 decisions per second) or every few timestamps (for example one decision per second is enough).
The maximum number of slices in the training phase is 30 (out of a total of 60 slices); the fewer slices in the training phase, the better.
Each student will work with one of the 8 subjects:

```
ID = 313131 #your ID number
s=ID%8+1 # subject index from 1 to 8
```

Description [5]

You are responsible of your choices. **Report due by Wednesday January 17th 2023, 11.59 PM**.

Table of Contents

Read the data [1]

- In DropBox you find the zipped file with the time series. You can download the data from `https://archive.ics.uci.edu/ml/datasets/Daily+and+Sports+Activities`.
- Copy the file on your PC, extract the folders and files.
- **NOTE: do NOT include the data when you upload the report and the Python code**.
- In DropBox you also find a Python script that allows you to generate a Pandas dataframe that includes the selected slices of the selected activities of the selected subject.

Read the data [2]

```
def generateDF(filedir,colnames,patients,activities,slices):
    # get the data from files for the selected patients
    # and selected activities
    # concatenate all the slices
    # generate a pandas dataframe with an added column: activity
    x=pd.DataFrame()
    for pat in patients:
        for a in activities:
            subdir='a'+f"{a:02d}"+'/p'+str(pat)+'/'
            for s in slices:
                filename=filedir+subdir+'s'+f"{s:02d}"+'.txt'
                x1=pd.read_csv(filename,names=colnames)
                x1['activity']=a*np.ones((x1.shape[0],),dtype=int)
                x=pd.concat([x,x1], axis=0, join='outer',
                            ignore_index=True,
                            keys=None, levels=None, names=None,
                            verify_integrity=False,
                            sort=False, copy=True)
    return x
```

Read the data [3]

- Activities

```
actNames=[
    'sitting',  # 1
    'standing', # 2
    'lying on back',# 3
    'lying on right side', # 4
    'ascending stairs' , # 5
    'descending stairs', # 6
    'standing in an elevator still', # 7
    'moving around in an elevator', # 8
    'walking in a parking lot', # 9
    'walking on a treadmill with a speed of 4 km/h in flat', # 10
    'walking on a treadmill with a speed of 4 km/h in 15 deg inclined positi
    'running on a treadmill with a speed of 8 km/h', # 12
    'exercising on a stepper', # 13
    'exercising on a cross trainer', # 14
    'cycling on an exercise bike in horizontal positions', # 15
    'cycling on an exercise bike in vertical positions', # 16
    'rowing', # 17
    'jumping', # 18
    'playing basketball' # 19
   ]
```

Read the data [4]

- Short names for activities:

```
actNamesShort=[
      'sitting',   # 1
      'standing', # 2
      'lying.ba', # 3
      'lying.ri', # 4
      'asc.sta' , # 5
      'desc.sta', # 6
      'stand.elev', # 7
      'mov.elev', # 8
      'walk.park', # 9
      'walk.4.fl', # 10
      'walk.4.15', # 11
      'run.8', # 12
      'exer.step', # 13
      'exer.train', # 14
      'cycl.hor', # 15
      'cycl.ver', # 16
      'rowing', # 17
      'jumping', # 18
      'play.bb' # 19
      ]
```

Read the data [5]

- Sensors

```
sensNames=[
        'T_xacc', 'T_yacc', 'T_zacc',
        'T_xgyro','T_ygyro','T_zgyro',
        'T_xmag', 'T_ymag', 'T_zmag',
        'RA_xacc', 'RA_yacc', 'RA_zacc',
        'RA_xgyro','RA_ygyro','RA_zgyro',
        'RA_xmag', 'RA_ymag', 'RA_zmag',
        'LA_xacc', 'LA_yacc', 'LA_zacc',
        'LA_xgyro','LA_ygyro','LA_zgyro',
        'LA_xmag', 'LA_ymag', 'LA_zmag',
        'RL_xacc', 'RL_yacc', 'RL_zacc',
        'RL_xgyro','RL_ygyro','RL_zgyro',
        'RL_xmag', 'RL_ymag', 'RL_zmag',
        'LL_xacc', 'LL_yacc', 'LL_zacc',
        'LL_xgyro','LL_ygyro','LL_zgyro',
        'LL_xmag', 'LL_ymag', 'LL_zmag']
```

Read the data [6]

- Folder with data:

```
1 filedir='../data/'
```

In the given version, a folder e.g. `lab3` includes folder `data` and folder `surname_s313131_report_2_code`; folder `surname_s313131_report_2_code` includes file `data_visualization_forstudents.py`

Read the data [7]

- Settings:

```
ID=313131
s=ID%8+1
patients=[s]  # list of selected patients
activities=list(range(1,6)) #list of indexes of activities to plot
Num_activities=len(activities)
NAc=19 # total number of activities
actNamesSub=[actNamesShort[i-1] for i in activities] # short names of the s
sensors=list(range(9)) # list of sensors
sensNamesSub=[sensNames[i] for i in sensors] # names of selected sensors
Nslices=12 # number of slices to plot
#Ntot=60 #total number of slices
slices=list(range(1,Nslices+1))# first Nslices to plot
fs=25 # Hz, sampling frequency
samplesPerSlice=fs*5 # samples in each slice
```

In the example only the first 9 sensors and the first 5 activities are considered. In plots only the first 12 slices (60 seconds) are shown. **It is highly recommended that you see the time series associated with all the activities and possibly all the sensors**.

Table of Contents

“Cluster” analysis [1]

- In this lab we know what we would like K-Means to give us: each cluster should include exactly all the rows/samples of one and only one activity.
- It is therefore useful to start analyzing the data as they are, assuming that K-Means will then be able to correctly find the clusters.
- The cluster centroids can be obtained as the mean of the data for each activity. It is also interesting to find the standard deviations of the measures of each sensor for each activity, because this tells us how much the the points move around the mean (for each sensor/feature).

“Cluster” analysis [2]

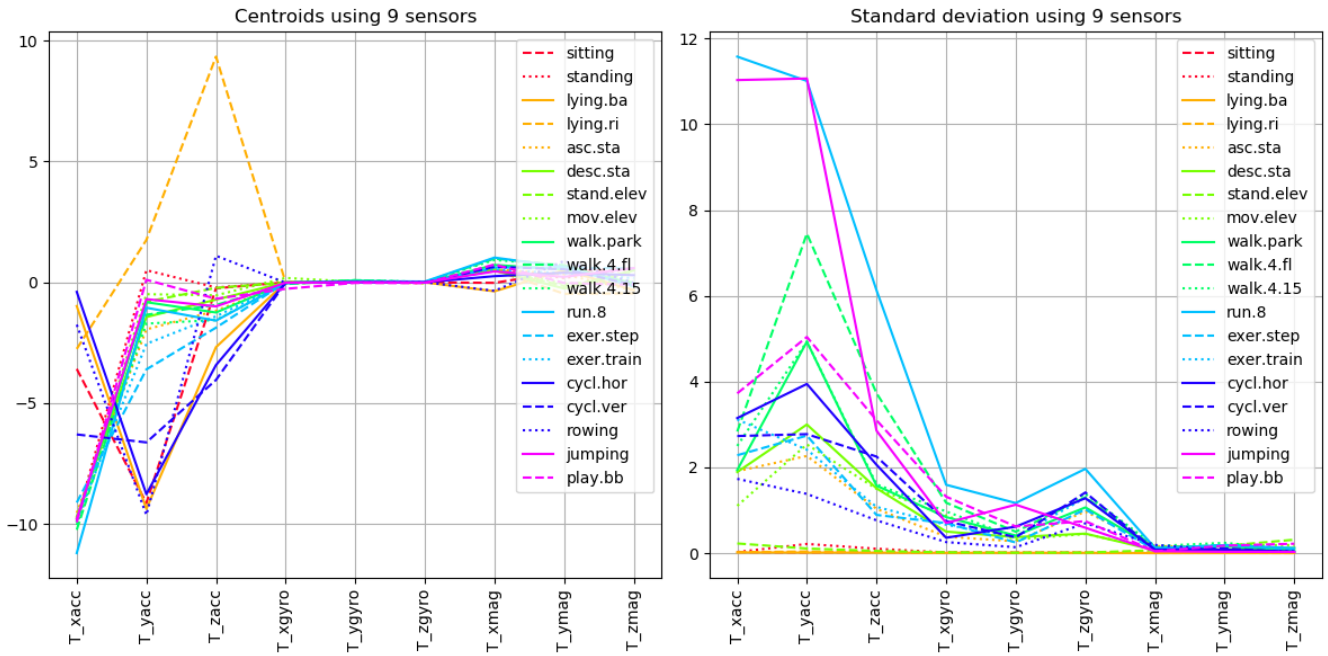
All the activities are here considered, but only 9 features are used:

```
#%% plot centroids and stand. dev. of sensor values
print('Number of used sensors: ',len(sensors))
centroids=np.zeros((NAc,len(sensors)))# centroids for all the activities
stdpoints=np.zeros((NAc,len(sensors)))# variance in cluster for each sensor
plt.figure(figsize=(12,6))
for i in range(1,NAc+1):
    activities=[i]
    x=generateDF(filedir,sensNamesSub,patients,activities,slices)
    x=x.drop(columns=['activity'])
    centroids[i-1,:]=x.mean().values
    plt.subplot(1,2,1)
    lines = plt.plot(centroids[i-1,:],label=actNamesShort[i-1])
    lines[0].set_color(cm(i//3*3/NAc))
    lines[0].set_linestyle(line_styles[i%3])
    stdpoints[i-1]=np.sqrt(x.var().values)
    plt.subplot(1,2,2)
    lines = plt.plot(stdpoints[i-1,:],label=actNamesShort[i-1])
    lines[0].set_color(cm(i//3*3/NAc))
    lines[0].set_linestyle(line_styles[i%3])
plt.subplot(1,2,1)
plt.legend(loc='upper right')
plt.grid()
```

“Cluster” analysis [3]

```
23 plt.title('Centroids using '+str(len(sensors))+' sensors')
24 plt.xticks(np.arange(x.shape[1]),list(x.columns),rotation=90)
25 plt.subplot(1,2,2)
26 plt.legend(loc='upper right')
27 plt.grid()
28 plt.title('Standard deviation using '+str(len(sensors))+' sensors')
29 plt.xticks(np.arange(x.shape[1]),list(x.columns),rotation=90)
30 plt.tight_layout()
```

“Cluster” analysis [4]



It seems that jumping and running at 8 km/h imply large variations of the values measured by the torso accelerometer, whereas the torso gyroscope mean and standard deviation values seem not to depend that much on the activity.

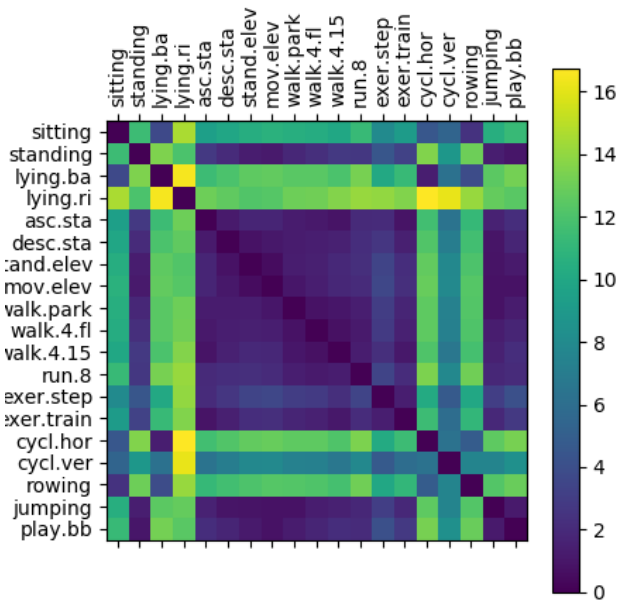
“Cluster” analysis [5]

- Let us now find the distance among the centroids. The centroids should be well separated, so that the subsequent classification can be more easily and correctly accomplished.

```
#%% between centroids distance
d=np.zeros((NAc,NAc))
for i in range(NAc):
    for j in range(NAc):
        d[i,j]=np.linalg.norm(centroids[i]-centroids[j])

plt.matshow(d)
plt.colorbar()
plt.xticks(np.arange(NAc),actNamesShort,rotation=90)
plt.yticks(np.arange(NAc),actNamesShort)
```

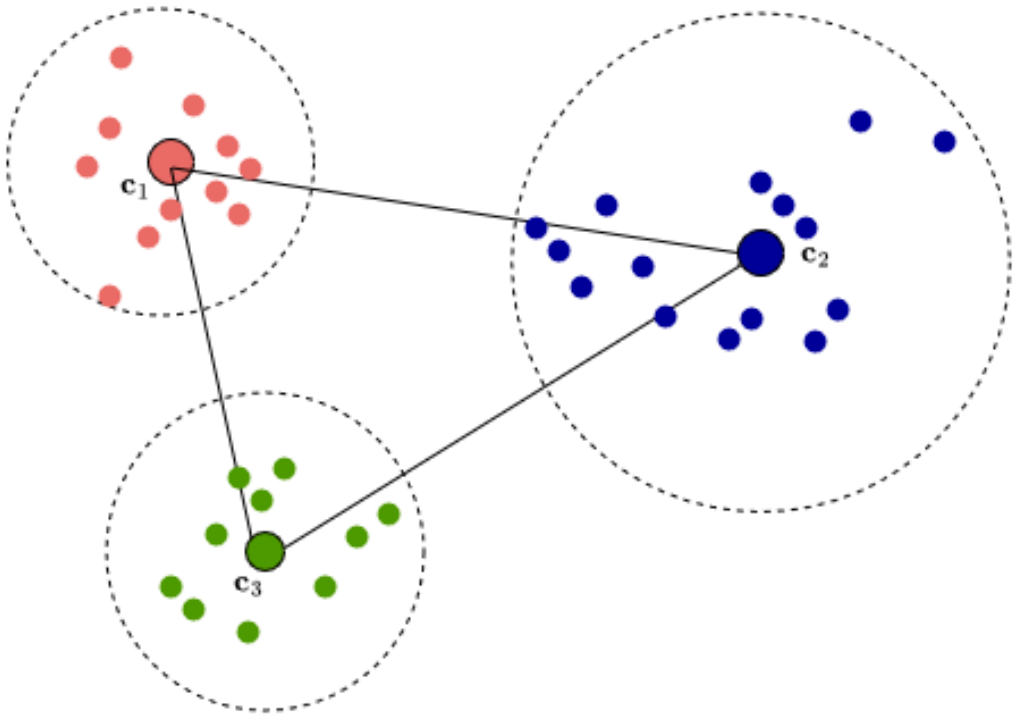
“Cluster” analysis [6]



It seems that it will be very difficult to distinguish the activities from acending the stairs to exercising on a cross trainer, at least using the data as they are and just 9 features, because the distances among these centroids are very small.

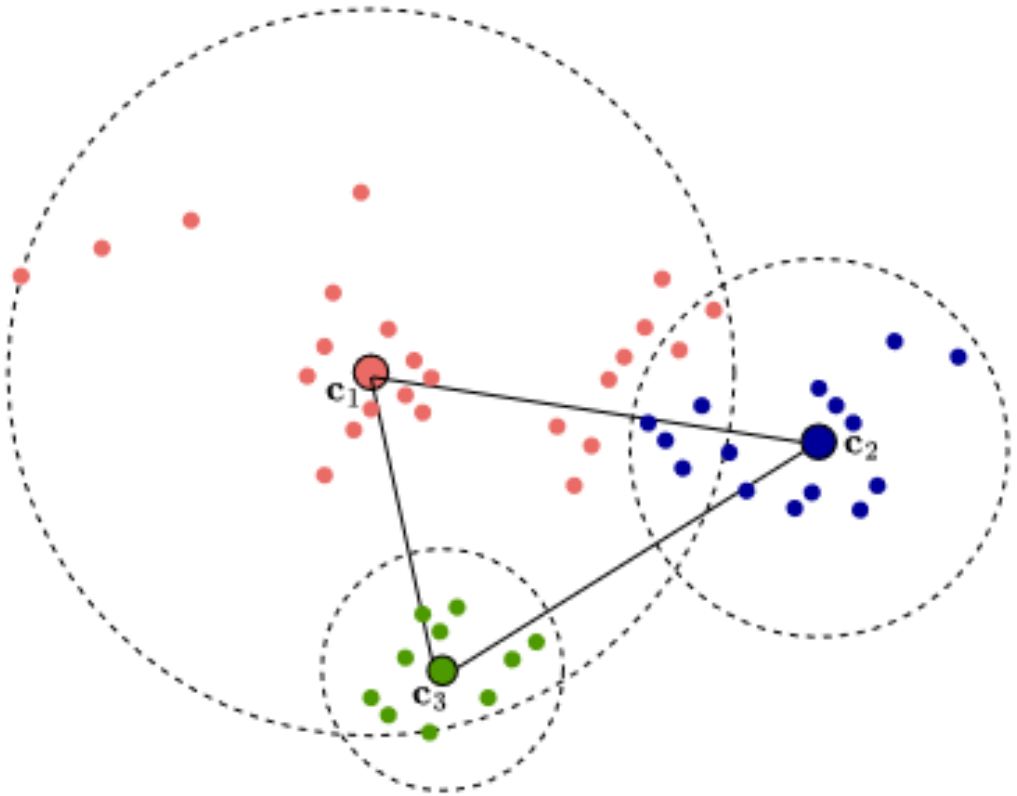
“Cluster” analysis [7]

Desired case: clusters are far apart and points are close to their centroids. K-Means can work correctly.



“Cluster” analysis [8]

Undesired case: points are very spread and distant from centroids.
K-Means cannot work correctly (it finds other centroids and clusters).



"Cluster" analysis [9]

- For each activity find the minimum distance between the corresponding centroid and the other centroids:

```
dd=d+np.eye(NAc)*1e6# remove zeros on the diagonal (distance of centroid fro
dmin=dd.min(axis=0)# find the minimum distance for each centroid
```

- Find the average distance of the points of a cluster from their centroid. The average square distance is

$$d^2 = \frac{1}{N} \sum_{n=1}^{N} \sum_{f=1}^{F} (\mathbf{x}_{nf} - \mathbf{c}_f)^2$$

where N is the number of points in the cluster, F is the number of features/columns, $\mathbf{c}$ is the centroid, $\mathbf{x}_n$ is the n-th point in the cluster, $\mathbf{x}_{nf}$ is the value of the f-th feature of the n-th point. Note that

$$d^2 = \sum_{f=1}^{F} \frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}_{nf} - \mathbf{c}_f)^2 = \sum_{f=1}^{F} \mathrm{var}_f$$

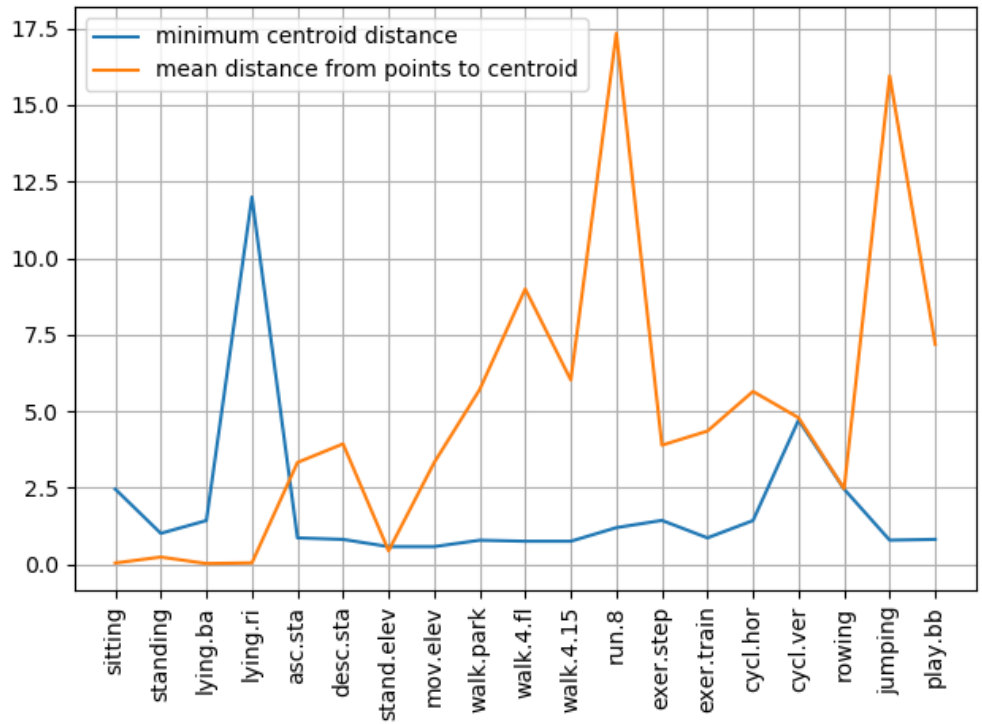
“Cluster” analysis [10]

where var_f is the variance of feature f in the cluster. Therefore, the average distance can be evaluated as

$$d = \sqrt{\sum_{f=1}^{F} \text{var}_f}$$

```
1 dpoints=np.sqrt(np.sum(stdpoints**2,axis=1))
```

“Cluster” analysis [11]



If the mean distance of points from the centroid is higher than the minimum distance between the centroid and the other centroids, then we are in the undesired case and the corresponding activity cannot be detected. Only the first 4 activities can be easily detected.

Table of Contents

To do [1]

1. For your specific subject (that depends on your ID number), have a look at all the plots, all the activities, possibly all the features/sensors/axis.
2. Work on the time series/data, through data processing, to improve the plot in slide 26 (mean distance of points in the cluster should be less than the minimum distance between the cluster centroid and all the other centroids).
3. Select the number of training slices, select **all the activities** and generate the training and test datasets. If, for example, you want to use 10 slices for training, then your training dataset will have $10 \times 125 \times 19$ rows, the test dataset will have $50 \times 125 \times 19$ rows (at least with the current initial settings, with no data processing). It is not necessary to shuffle the data, since K-Means does not take advantages of data order.
4. Run Scikit-Learn K-Means algorithm (fit the training dataset) and run it on the test dataset (predict). Choose appropriate parameters in the Scikit-Learn K-Means call.

To do [2]

5. **Note:** the **index of clusters at the output of K-Means is random**. Not necessarily activity 1 corresponds to cluster 1, activity 1 might correspond to cluster 10. Find the **mapping** between K-Means cluster indexes and activities, so that you can correctly remap cluster indexes to activities.
6. For both training and test datasets, find the confusion matrix that gives

$$P(i|j) = P(\text{detected activity is } i|\text{the true activity is } j)$$

in row j column i, for $i = 1, \ldots, 19$ and $j = 1, \ldots, 19$. Index i corresponds to activity i, index j corresponds to activity j.

To do [3]

7. For both training and test datasets, find the accuracy:

$$\text{Accuracy} = \sum_{i=1}^{19} P(i|i)P(i)$$

where $P(i)$ is the probability of activity i, $P(i) = 1/19$. These values must be written in the report.

8. Find the accuracy of each activity and write the corresponding table in the report. Find if two activities are easily confused, etc.

9. With an appropriate data processing, you should get an accuracy around 0.85-0.95 for both training and test data, these values change with the subject. If you cannot, try with other time series processing techniques. If you cannot find a solution turn in your best solution.

10. Write the report, **6 pages** (same rules as for report 1), in which you exactly exlain what you did, so that another researcher can repeat your experiment and obtain the same results.

To do [4]

11. The report must include at least the plot of the confusion matrix, the accuracies for each of the 19 activities, the overall accuracy for training and test datasets.
12. **NOTE: do NOT include the data when you upload the report and the Python code**.
13. It is forbidden to **copy** figures or sentences from these slides, from the lecture slides, from Wikipedia, from the web, etc (this is **plagiarism**). You must write your own sentences with your own words. If you need a figure, you produce it.
14. You **cannot discuss** the method you decided to adopt with the other students, each student must work on his/her own. If the methods (and/or the report) of two students are too similar, the grade will be divided by 2 (if you find a solution, don't tell it to other students otherwise you'll get half of the correct grade, do not trust your colleagues).

To do [5]

Start being men and women responsible of their actions, not cheating children. You do not learn if you cheat, and you are paying for learning (both time and money). You will not be able to cheat in working activities: your colleagues will have their own job to do, they will not help you.