Machine Learning Project

Human Action Recognition Using Inertial Sensors

1 Project presentation

The objective of this project is to implement a system for human action recognition using inertial sensors. Figure I illustrates the recognition of actions with these sensors implemented in a smartphone. The inertial sensors embedded inside the smartphone worn by the person, namely the accelerometer, gyroscope, and magnetometer, allow for the collection of inertial signals. These signals will then be fed into an action recognition system to identify the executed action in real-time. In the example shown in Figure I, the system is expected to output the label "walking" for the action ; otherwise, it will be considered a classification error.

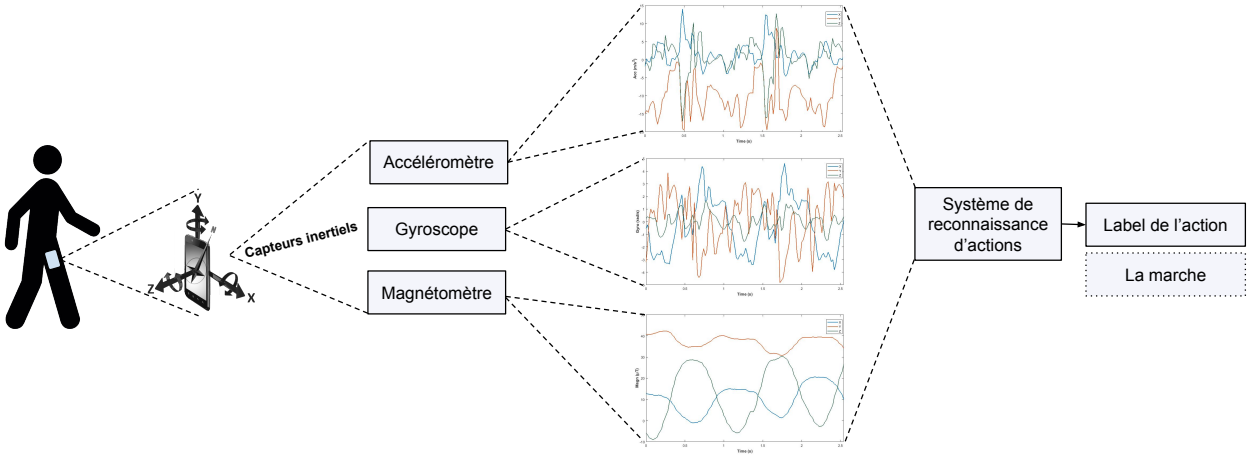


FIGURE I – Recognition of human action using a smartphone.

2 Task

The work to be carried out is divided into several steps that are characteristic of a machine learning project : loading and understanding the data, extracting relevant information (attributes), preparing the data for the training phase, training a classification model, testing the model to measure and analyze its performance. After successfully completing these various steps, the saved model is ready to be deployed.

The steps are described in the following sections. It is advisable to follow the indicated order. For each step, explanations and indications of functions to use in different libraries are provided. Your task is to understand the explanations, explore the documentation of the mentioned functions, and write the code to perform each step. Remember to check that your code produces the expected results.

At the end of the project, you will need to provide your commented code as well as a clear and concise report written in the style of a study report.

3 Development environment

This project will be carried out in Python, a highly popular programming language in artificial intelligence and machine learning. If you haven't installed Python on your machine yet, you can install Anaconda, a free and open-source distribution of the Python language (visit the Anaconda installation page). Upon installation, Anaconda will install Python along with a variety of IDEs and packages (see the list of packages).

Anaconda provides a package management tool called Conda, which will make it easy to update and install the libraries needed for this project. Conda also allows you to create a new virtual environment : a directory containing a specific collection of Conda packages that you have installed. To do this, go to the 'Environments' tab, create a new virtual environment by clicking 'Create,' and then install or update the packages you need.

List of libraries you will need for this project : numpy, pandas, matplotlib, SciPy, Scikit-Learn.

4 Database

The database used in this project (see Figure II) contains 27 different actions : (1) right-arm sweep to the left, (2) right-arm sweep to the right, (3) right-hand waving, (4) clapping, (5) right-arm throwing, (6) crossing arms on the chest, (7) basketball shooting, (8) drawing an 'x', (9) drawing a circle with the right hand (clockwise), (10) drawing a circle with the right hand (counterclockwise), (11) drawing a triangle, (12) right-hand bowling, (13) boxing, (14) baseball swing, (15) tennis swing, (16) arm flexion (both arms), (17) tennis serve, (18) pushing, (19) knocking on the door, (20) object catching, (21) pick up and throw, (22) running in place, (23) walking, (24) standing up, (25) sitting down, (26) left foot forward, (27) squatting. Eight subjects participated in this experiment, including 4 men and 4 women.

Each subject repeated each action 4 times. After removing three corrupted sequences, the database comprises a total of 861 sequences. The data were stored using the MATLAB development environment.

5 Data collection procedure

To collect data, wearable sensors known as Microelectromechanical Systems (MEMS) were used, specifically inertial MEMS as presented in Figure III1. This device contains two motion sensors : the accelerometer, which calculates static and dynamic accelerations of a device, and the gyroscope, which calculates angular velocity. The signals are collected along three perpendicular axes (x, y, z). The sampling frequency for both sensors is $50Hz$.

In the database, the inertial MEMS was worn on the right wrist of the subject for actions that primarily engage the arms (actions 1 to 21), or on the right thigh for actions that engage the legs (actions 22 to 27).

6 Data loading

Download the database in zip format IMU.zip into the Project folder on MyLearningSpace. In this folder, you will find files with the extension 'mat' (for MATLAB). The files are named according to the format 'ai_sj_tk_inertial.mat,' where 'ai' represents action number i, 'sj' represents subject number j, and 'tk' signifies trial number k.

↪ After downloading the database, write a function called "load_data" that allows loading the data into Python (you can use the loadmat function from the Scipy library). The function should return a 'dataframe' organized as follows :

- Columns 0-2 : contain accelerometer data along the three axes.
- Columns 3-5 : contain gyroscope data along the three axes.
- Column 6 : contains the subject identifier (1 to 8).
- Column 7 : contains the trial identifier (1 to 4).
- Column 8 : contains the identifier (label) of the action (1 to 27).

The result should look like Figure IV. The Python method **listdir(path)** from the os module returns a list containing the names of the files in the directory specified by path.

↪ Write a function called "plot_signal" that allows plotting the three signals (x, y, z) of a sensor corresponding to an action. This function takes 5 inputs :

- The dataframe (returned by the “load_data” function).
- The sensor (1 : accelerometer, 2 : gyroscope).

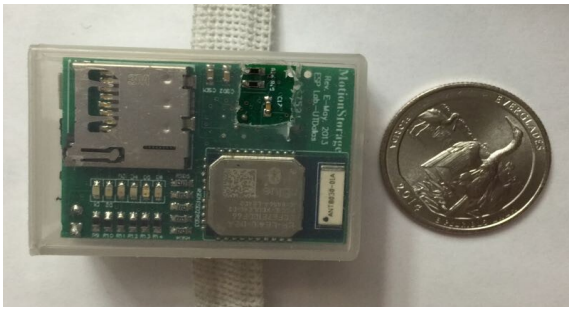
FIGURE II – Actions humaines contenues dans la base de données.

- The action number.
- The subject number.
- The trial number.

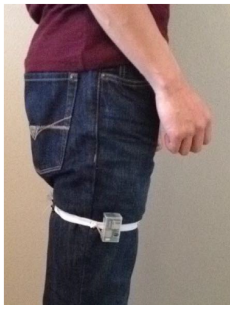
You will need the 'matplotlib' library to plot a signal. Click here for an example.

7 Attribute extraction

To describe each human action, a list of attributes is extracted from the signals. In this project, we will opt for statistical attribute extraction. Here is a list of attributes that can be



(1) MEMES inertiel.



(2) Position sensors

FIGURE III – Sensors

	0	1	2	3	4	5	subject	experience	action
0	-0.95166	-0.41626	-0.09106	-18.19847	-8.67176	8.61069	1	1	10
1	-0.94849	-0.42554	-0.11865	-18.71756	-7.84733	6.44275	1	1	10
2	-0.94043	-0.42602	-0.14087	-21.40458	-5.67939	2.22901	1	1	10
3	-0.94482	-0.50757	-0.19141	-23.38931	-2.22901	-2.83969	1	1	10
4	-0.95459	-0.63696	-0.30737	-25.12977	2.13741	-6.16794	1	1	10
5	-0.94824	-0.70337	-0.37012	-31.81679	8.42748	-11.69466	1	1	10
6	-0.96387	-0.73828	-0.36133	-37.61832	16.82443	-23.11450	1	1	10
7	-0.96045	-0.79248	-0.38013	-33.49618	26.38168	-40.88550	1	1	10
8	-1.01392	-0.89014	-0.43823	-26.65649	35.14504	-61.92366	1	1	10
9	-1.01099	-0.97095	-0.46924	-29.67939	46.19847	-78.99237	1	1	10
10	-1.00513	-0.98682	-0.45288	-47.29771	58.07634	-96.42748	1	1	10
11	-1.02466	-0.94165	-0.43384	-60.27481	68.45802	-113.03817	1	1	10
12	-1.02978	-0.90478	-0.42822	-60.61069	76.67176	-127.93893	1	1	10
13	-1.03980	-0.92993	-0.44141	-58.59542	83.57252	-141.58779	1	1	10
14	-1.03662	-0.93164	-0.37598	-58.87023	89.92366	-151.05344	1	1	10
15	-1.02734	-0.93457	-0.33789	-61.31298	94.22901	-160.21374	1	1	10
16	-0.98340	-0.88989	-0.32324	-66.32061	96.70229	-169.34351	1	1	10
17	-0.92114	-0.82446	-0.31274	-67.87786	97.19084	-173.98473	1	1	10
18	-0.85425	-0.75073	-0.29565	-66.50382	96.15267	-173.70992	1	1	10
19	-0.77661	-0.67651	-0.25537	-60.45802	93.40458	-170.19847	1	1	10

FIGURE IV – Example of loaded data.

extracted : mean, root mean square, standard deviation, median, quartiles, skewness coefficient, kurtosis, entropy, energy, maxima and minima, correlation coefficient, histogram, zero-crossings, number of peak occurrences.

↪ Write a function called "feature_extraction" that takes the created dataframe as input and calculates a feature vector for each action.

Note : If you choose, for example, mean and standard deviation as attributes, these must be calculated for each signal (columns 0-5 in Figure IV). The feature vector for each action will therefore be of dimension 12 (6 signals $\times$ 2 attributes). It is represented as follows :
$V = [mean(A_x),\ std(A_x),\ mean(A_y),\ std(A_y),\ mean(A_z),\ std(A_z),\ mean(G_x),$
$std(G_x),\ mean(G_y),\ std(G_y),\ mean(G_z),\ std(G_z)]$

NB : The **dataframe.groupby()** function allows splitting the dataframe data into groups based on certain criteria.

8 Data preparation

In order to prepare the data (calculated attributes) for the classification step, they need to be divided into two groups : training data and testing data. In this project, the training data should contain the calculated attributes for subjects $[1, 3, 5, 7]$, and the testing data for subjects $[2, 4, 6, 8]$.

An important step (but not necessary, it depends on the calculated attributes) is data normalization. To apply this, calculate the vector of means and standard deviations on the training attributes (the dimensions of the vectors should be equal to the number of calculated attributes). Then subtract the mean vector from both the training and testing data and divide by the standard deviation.

↪ Create a function that performs this task. The function should return four elements : training data, training labels, testing data, testing labels.

9 Training a Classification Model

To initiate the classification of human actions, we will first train a model on the training data. For this, we will use "Scikit-learn", a free Python library for machine learning.

↪ Install "Scikit-learn" using the command : pip install scikit-learn

↪ Apply some classifiers (the most well-known : neural network, support vector machine, k-nearest neighbors, decision trees, Naïve Bayes). For this, refer to the page scikit-

learn.

Example of applying the decision tree classifier :

```
from sklearn import tree                    #Import descison tree
data = [[0, 0], [1, 1]]                     #Training data (feature extracted)
labels = [0, 1]                             #labels (in our project action 1-27)
clf = tree.DecisionTreeClassifier()         #Initialize our classifier
clf = clf.fit(data, labels)                 #Train our classifier
```

↪ Display the decision tree obtained with the decision tree classifier (you can use the plot_tree function from the sklearn.tree module).

↪ Display the decision tree obtained with the decision tree classifier in the form of nested rules (you can use the export_text function).

10 Model Testing

To test our model, we will use the testing data (which is independent of the data used in the training phase).

Example of application for the decision tree :

```
clf.predict([[2., 2.]])  #Predict action label from feature vector [2, 2]
```

↪ Write the code that allows using the testing data as input to the classifier (or classifiers if you have trained multiple). The result will be stored as a vector containing the labels of the actions.

11 Data Analysis

To estimate the overall recognition performance of our model, various metrics can be calculated, the most popular being *precision*, *recall*, *F-score*, and *accuracy*.

↪ Write a function that calculates these metrics ; you can use the classification_report" available in sklearn" (see classification report). Comment on the results obtained.

The quality metrics mentioned above provide an overall score that quickly indicates whether our system is performing well or not. To get more details on the system's quality (score obtained for each action, most confused actions, etc.), the confusion matrix is calculated.

↪ Determine the confusion matrix using confusion_matrix" available in sklearn" (see confusion matrix). Comment on the results obtained.