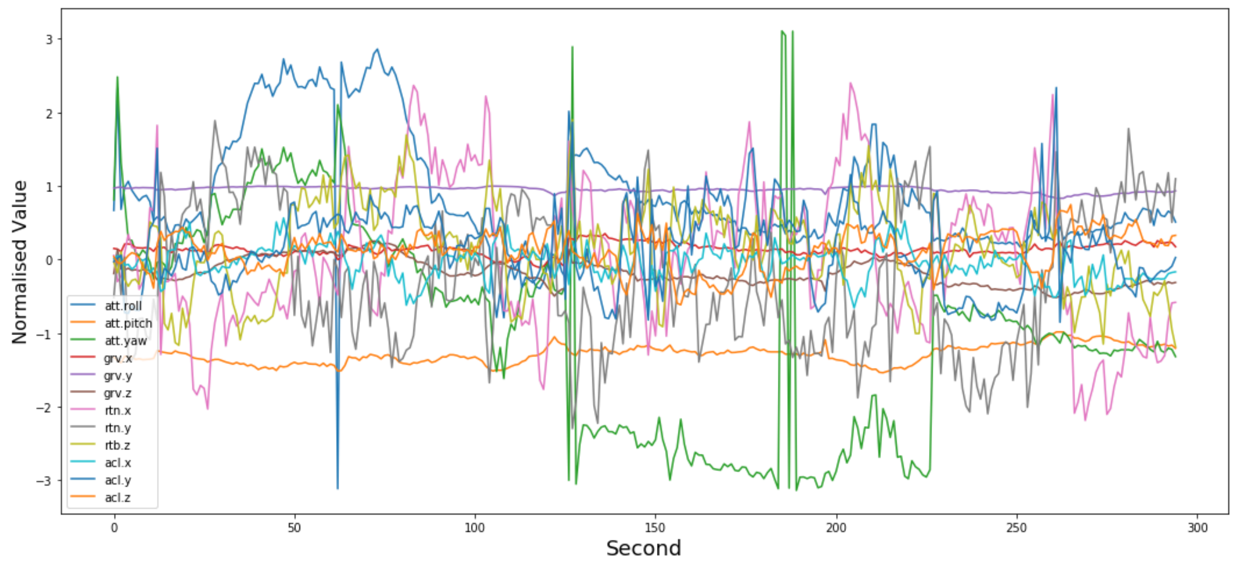
mmalekzadeh / **motion-sense** Public

MotionSense Dataset for Human Activity and Attribute Recognition (time-series data generated by smartphone's sensors: accelerometer and gyroscope) (PMC Journal) (IoTDI'19)

www.sciencedirect.com/science/article/pii/s1574119220300201

MotionSense Dataset

This dataset includes time-series data generated by accelerometer and gyroscope sensors (attitude, gravity, userAcceleration, and rotationRate). It is collected with an iPhone 6s kept in the participant's front pocket using SensingKit which collects information from Core Motion framework on iOS devices. All data collected in 50Hz sample rate. A total of 24 participants in a range of gender, age, weight, and height performed 6 activities in 15 trials in the same environment and conditions: downstairs, upstairs, walking, jogging, sitting, and standing. With this dataset, we aim to look for *personal attributes fingerprints* in time-series of sensor data, i.e. attribute-specific patterns that can be used to infer gender or personality of the data subjects in addition to their activities.



Time-series correspond to Walking activity of data subject(code 3). There are 12-features.

Some Notes:

- If you are here for the paper "Privacy and Utility Preserving Sensor-Data Transformations", please look at `pmc_xxx` and `tutorial` folders.
- If you are training deep neural networks on sensor data, you may find our recent work in the following link useful: https://github.com/mmalekzadeh/dana

Download

The MotionSense dataset is publicly available in the current repository and also in the Queen Mary University of London's repository as a backup.

> There is also a Kaggle version: https://www.kaggle.com/malekzadeh/motionsense-dataset

Citation

If you find this dataset useful to your research, please cite one of the following papers:

```
@inproceedings{Malekzadeh:2019:MSD:3302505.3310068,
author = {Malekzadeh, Mohammad and Clegg, Richard G. and Cavallaro, Andrea and Haddadi,
title = {Mobile Sensor Data Anonymization},
booktitle = {Proceedings of the International Conference on Internet of Things Design a
series = {IoTDI '19},
year = {2019},
isbn = {978-1-4503-6283-2},
```

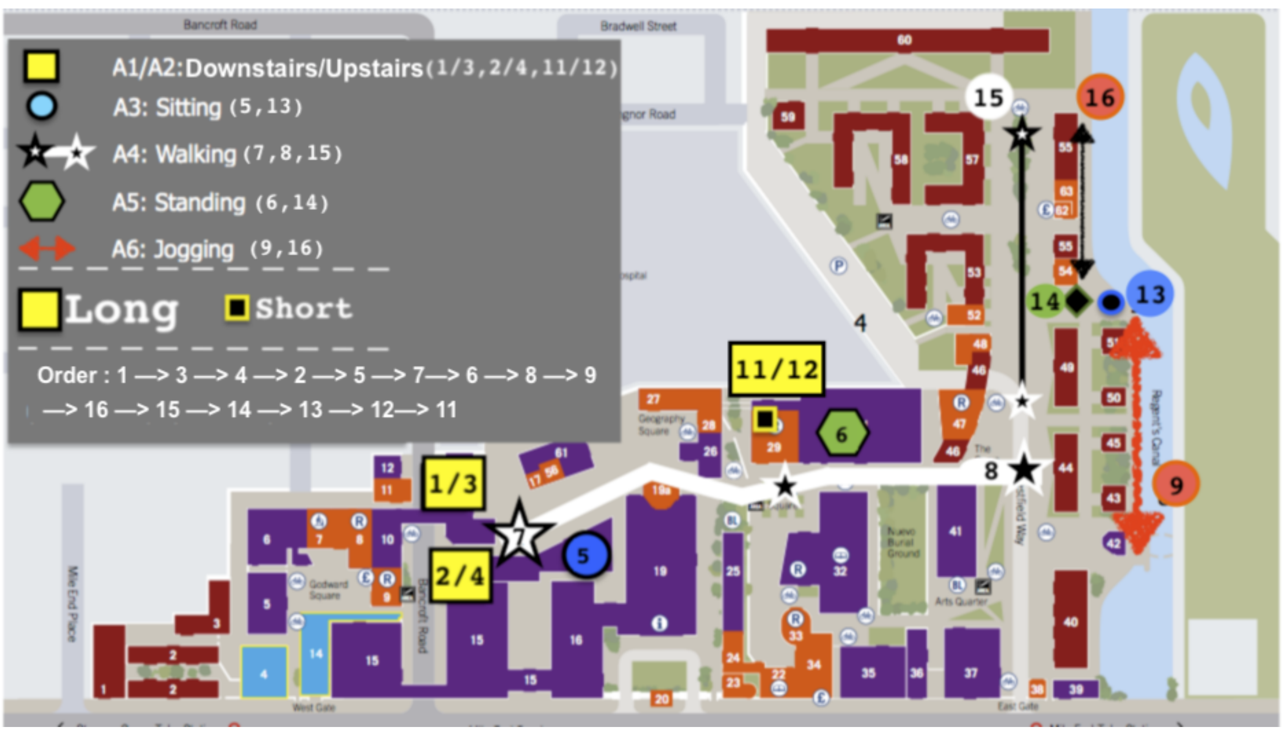
```
location = {Montreal, Quebec, Canada},
pages = {49--58},
numpages = {10},
url = {http://doi.acm.org/10.1145/3302505.3310068},
doi = {10.1145/3302505.3310068},
acmid = {3310068},
publisher = {ACM},
address = {New York, NY, USA},
keywords = {adversarial training, deep learning, edge computing, sensor data privacy, t
}
```

```
@inproceedings{Malekzadeh:2018:PSD:3195258.3195260,
 author = {Malekzadeh, Mohammad and Clegg, Richard G. and Cavallaro, Andrea and Haddadi
 title = {Protecting Sensory Data Against Sensitive Inferences},
 booktitle = {Proceedings of the 1st Workshop on Privacy by Design in Distributed Syste
 series = {W-P2DS'18},
 year = {2018},
 isbn = {978-1-4503-5654-1},
 location = {Porto, Portugal},
 pages = {2:1--2:6},
 articleno = {2},
 numpages = {6},
 url = {http://doi.acm.org/10.1145/3195258.3195260},
 doi = {10.1145/3195258.3195260},
 acmid = {3195260},
 publisher = {ACM},
 address = {New York, NY, USA},
 keywords = {Activity Recognition, Machine Learning, Privacy, Sensor Data, Time-Series
}
```

Dataset Description

Scenario

For each participant, the study had been commenced by collecting their demographic (age and gender) and physically-related (height and weight) information. Then, we provided them with a dedicated smartphone (iPhone 6) and asked them to store it in their trousers' front pocket during the experiment. All the participant were asked to wear flat shoes. We then asked them to perform 6 different activities (walk downstairs, walk upstairs, sit, stand and jogging) around the Queen Mary University of London's Mile End campus. For each trial, the researcher set up the phone and gave it to the current participants, then the researcher stood in a corner. Then, the participant pressed the start button of [Crowdsense app](#) and put it in their trousers' front pocket and performed the specified activity. We asked them to do it as natural as possible, like their everyday life. At the end of each trial, they took the phone out of their pocket and pressed the stop button. The exact places and routes for running all the activities are shown in the illustrative map in the following Figure.



As we can see, there are 15 trials:

1. Long trials: those with number 1 to 9 with around 2 to 3 minutes duration.
2. Short trials: those with number 11 to 16 that are around 30 seconds to 1 minutes duration.

Data Subjects

There are 24 data subjects. Here we summarized their information:

README.md

Code	Weight (kg)	Height (cm)	Age (years)	Gender (F:0,M:1)
2	72	[illegible]	[illegible]	1
3	48	161	28	0
4	90	176	31	1
5	48	164	23	0
6	76	180	28	1
7	62	175	30	0
8	52	161	24	0
9	93	190	32	1
10	72	164	31	0
11	70	178	24	1
12	60	167	33	1
13	60	178	33	1
14	70	180	35	1
15	70	185	33	1
16	96	172	29	0
17	76	180	26	1
18	54	164	26	0
19	78	164	28	0
20	88	180	25	1
21	52	165	24	1
22	100	186	31	1
23	68	170	25	0
24	74	173	18	0

Folders (and Features)

There three different folders. Usually, you just need the **folder (A)** (DeviceMotion), because this folder includes everything that can be captured by both Accelerometer and Gyroscope. However, we also have data captured by these two sensors separately in the folder (B) and (C).

(A) DeviceMotion_data

This folder contains time-series collected by both Accelerometer and Gyroscope for all 15 trials. For every trial we have a multivariate time-series, like this:

index	attitude.roll	attitude.pitch	attitude.yaw	gravity.x	gravity.y	gravit
0	-2.544349	-1.250641	2.175416	-0.176977	0.949187	0.260
1	-2.524075	-1.187355	2.047589	-0.21661	0.927383	0.305
2	-2.534324	-1.141923	1.990077	-0.237286	0.909435	0.341
3	-2.564504	-1.098202	1.960054	-0.248344	0.89039	0.381
...	...	...	...	...	...	...

Thus, we have time-series with 12 features:

1. attitude.roll
2. attitude.pitch
3. attitude.yaw
4. gravity.x
5. gravity.y
6. gravity.z
7. rotationRate.x
8. rotationRate.y
9. rotationRate.z
10. userAcceleration.x
11. userAcceleration.y
12. userAcceleration.z

For more information, please read this page: CMDeviceMotion

> The accelerometer measures the sum of two acceleration vectors: gravity and user acceleration. User acceleration is the acceleration that the user imparts to the device. Because Core Motion is able to track a device's attitude using both the gyroscope and the accelerometer, it can differentiate between gravity and user acceleration. A CMDeviceMotion object provides both measurements in the gravity and userAcceleration properties.

(B) Accelerometer_data

Here we just have data reported by **Accelerometer** sensor. Thus, there are just three features correspond to 3 different axes:

1. x
2. y
3. z

(C) Gyroscope_data

Here we just have data reported by **Gyroscope** sensor. Thus, there are again just three features correspond to 3 different axes:

1. x
2. y
3. z

Labels

There are 6 different labels:

1. **dws**: downstairs
2. **ups**: upstairs
3. **sit**: sitting
4. **std**: standing
5. **wlk**: walking
6. **jog**: jogging

A Code to Build a Labeled Time-Series from data into a Pandas DataFrame

```
import numpy as np
import pandas as pd

def get_ds_infos():
    """
    Read the file includes data subject information.

    Data Columns:
    0: code [1-24]
    1: weight [kg]
    2: height [cm]
    3: age [years]
    4: gender [0:Female, 1:Male]

    Returns:
        A pandas DataFrame that contains inforamtion about data subjects' attributes
    """

    dss = pd.read_csv("data_subjects_info.csv")
    print("[INFO] -- Data subjects' information is imported.")

    return dss

def set_data_types(data_types=["userAcceleration"]):
    """
    Select the sensors and the mode to shape the final dataset.

    Args:
        data_types: A list of sensor data type from this list: [attitude, gravity, rota

    Returns:
        It returns a list of columns to use for creating time-series from files.
    """
    dt_list = []
    for t in data_types:
        if t != "attitude":
            dt_list.append([t+".x",t+".y",t+".z"])
        else:
            dt_list.append([t+".roll", t+".pitch", t+".yaw"])

    return dt_list


def creat_time_series(dt_list, act_labels, trial_codes, mode="mag", labeled=True):
    """
    Args:
        dt_list: A list of columns that shows the type of data we want.
        act_labels: list of activites
        trial_codes: list of trials
        mode: It can be "raw" which means you want raw data
```

```
        for every dimention of each data type,
        [attitude(roll, pitch, yaw); gravity(x, y, z); rotationRate(x, y, z); userAccel
        or it can be "mag" which means you only want the magnitude for each data type:
        labeled: True, if we want a labeld dataset. False, if we only want sensor value

    Returns:
        It returns a time-series of sensor data.

    """
    num_data_cols = len(dt_list) if mode == "mag" else len(dt_list*3)

    if labeled:
        dataset = np.zeros((0,num_data_cols+7)) # "7" --> [act, code, weight, height, a
    else:
        dataset = np.zeros((0,num_data_cols))

    ds_list = get_ds_infos()

    print("[INFO] -- Creating Time-Series")
    for sub_id in ds_list["code"]:
        for act_id, act in enumerate(act_labels):
            for trial in trial_codes[act_id]:
                fname = 'A_DeviceMotion_data/'+act+'_'+str(trial)+'/sub_'+str(int(sub_i
                raw_data = pd.read_csv(fname)
                raw_data = raw_data.drop(['Unnamed: 0'], axis=1)
                vals = np.zeros((len(raw_data), num_data_cols))
                for x_id, axes in enumerate(dt_list):
                    if mode == "mag":
                        vals[:,x_id] = (raw_data[axes]**2).sum(axis=1)**0.5
                    else:
                        vals[:,x_id*3:(x_id+1)*3] = raw_data[axes].values
                    vals = vals[:,:num_data_cols]
                if labeled:
                    lbls = np.array([[act_id,
                            sub_id-1,
                            ds_list["weight"][sub_id-1],
                            ds_list["height"][sub_id-1],
                            ds_list["age"][sub_id-1],
                            ds_list["gender"][sub_id-1],
                            trial
                           ]]*len(raw_data))
                    vals = np.concatenate((vals, lbls), axis=1)
                dataset = np.append(dataset,vals, axis=0)
    cols = []
    for axes in dt_list:
        if mode == "raw":
            cols += axes
        else:
            cols += [str(axes[0][:-2])]

    if labeled:
```

```
        cols += ["act", "id", "weight", "height", "age", "gender", "trial"]

    dataset = pd.DataFrame(data=dataset, columns=cols)
    return dataset
#________________________________


ACT_LABELS = ["dws","ups", "wlk", "jog", "std", "sit"]
TRIAL_CODES = {
    ACT_LABELS[0]:[1,2,11],
    ACT_LABELS[1]:[3,4,12],
    ACT_LABELS[2]:[7,8,15],
    ACT_LABELS[3]:[9,16],
    ACT_LABELS[4]:[6,14],
    ACT_LABELS[5]:[5,13]
}

## Here we set parameter to build labeld time-series from dataset of "(A)DeviceMotion_c
## attitude(roll, pitch, yaw); gravity(x, y, z); rotationRate(x, y, z); userAcceleratic
sdt = ["attitude", "userAcceleration"]
print("[INFO] -- Selected sensor data types: "+str(sdt))
act_labels = ACT_LABELS [0:4]
print("[INFO] -- Selected activites: "+str(act_labels))
trial_codes = [TRIAL_CODES[act] for act in act_labels]
dt_list = set_data_types(sdt)
dataset = creat_time_series(dt_list, act_labels, trial_codes, mode="raw", labeled=True)
print("[INFO] -- Shape of time-Series dataset:"+str(dataset.shape))
dataset.head()
```

```
[INFO] -- Selected sensor data types: ['attitude', 'userAcceleration']
[INFO] -- Selected activites: ['dws', 'ups', 'wlk', 'jog']
[INFO] -- Data subjects' information is imported.
[INFO] -- Creating Time-Series
[INFO] -- Shape of time-Series dataset:(767660, 13)
```

	attitude.roll	attitude.pitch	attitude.yaw	userAcceleration.x	userAcceleration.y	userAcceleration.z	act	id	weight	height	age	gender	trial
0	1.528132	-0.733896	0.696372	0.294894	-0.184493	0.377542	0.0	0.0	102.0	188.0	46.0	1.0	1.0
1	1.527992	-0.716987	0.677762	0.219405	0.035846	0.114866	0.0	0.0	102.0	188.0	46.0	1.0	1.0
2	1.527765	-0.706999	0.670951	0.010714	0.134701	-0.167808	0.0	0.0	102.0	188.0	46.0	1.0	1.0
3	1.516768	-0.704678	0.675735	-0.008389	0.136788	0.094958	0.0	0.0	102.0	188.0	46.0	1.0	1.0
4	1.493941	-0.703918	0.672994	0.199441	0.353996	-0.044299	0.0	0.0	102.0	188.0	46.0	1.0	1.0

See also:

- For splitting this dataset into train_test: https://github.com/mmalekzadeh/sensplit

Some research papers that use MotionSense dataset:

Our Work

- DANA: Dimension-Adaptive Neural Architecture for Multivariate Sensor Data
- Privacy and Utility Preserving Sensor-Data Transformations
- Mobile Sensor Data Anonymization
- Protecting Sensory Data against Sensitive Inferences

Others

Please see this and this.

Releases

No releases published

Packages

No packages published

Contributors 2


mmalekzadeh Mohammad Malekzadeh


TuffDev Adam Martin

Languages

- **Jupyter Notebook** 99.8%
- **Python** 0.2%