ENGSCI 712 – Computational Techniques for Signal Processing

Assignment 2: Human Activity Recognition

Andreas W. Kempa-Liehr

Department of Engineering Science

Due date **Wednesday 7th October 10:00pm**
CSP-assignment-2.pdf, Rev. `c467bd3`

Outline

Human Activity Recognition (HAR) Assignment

In this assignment, you will

- Record acceleration signals for two different activities
- Load the recorded signals into a Jupyter notebook
- Visualize the recorded signals
- Preprocess the recorded signal
- Characerize the recorded signals using systematic time-series feature engineering
- Develop a machine learning algorithm for Human Activity Recognition
- Analyse the performance of your algorithm

Get your device ready

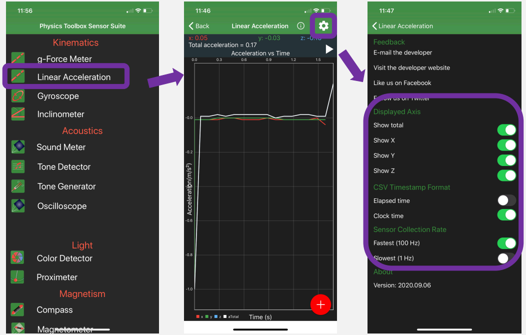
Install *Physics Toolbox Sensor Toolbox* on your device

- If you do not have access to Google's PlayStore or Apple's Appstore, you are free to install any other app, which allows to record and download acceleration signals from your device.

Configure App

Configure Linear Acceleration Settings

- Your device → Physics Toolbox Sensor Suite → Linear Acceleration →



Task 1 – Record acceleration signal for two different activities

- Use the record function  of the *Linear Acceleration* tab and record acceleration data for two different activities.
    - e.g. running and walking,
    - or walking upstairs / walking downstairs
- Each activity should be recorded for at least two minutes.
- Store one CSV file for each activity and share the files with your personal computer.
- Document your experiment by
    - describing the activities, and
    - specifying, where your device was located during the measurement.
    - This documentation should be added to a Markdown cell of the Juypter notebook, which is created in the following task.

Task 2 – Loading the raw data

- Create a Jupyter notebook using the virtual environment from the first assignment.
- Use the `pandas.read_csv()` function for leading the CSV files.
  - Hint: Read the documentation of `pandas.read_csv()` carefully.
  - You will need to parameterize the `pandas.read_csv()` function using the following parameters:
    - `parse_dates`, `skiprows`, `header`, `names`, `index_col`
- After loading the CSV files into a DataFrame variable `df`. The command `df.head()` should show a similar formatting to

	ax	ay	az	|a|
time				
2020-09-10 14:08:22.287	-0.66	0.64	1.75	1.97
2020-09-10 14:08:22.297	-0.61	0.40	1.18	1.38
2020-09-10 14:08:22.307	-0.47	0.12	0.51	0.70
2020-09-10 14:08:22.317	-0.40	0.01	0.01	0.40
2020-09-10 14:08:22.327	-0.31	-0.05	-0.48	0.57

Task 3 – Visualize the raw data

- Visualize the raw data using the `inline` plotting capabilities of Jupyter notebooks.
- Create one figure per activity.
- Describe the raw data:
  - What are the obvious differences between the signals recorded from the two different activities?
  - Can you identify transition times at the beginning time respectively at the end of each recording?

Task 4 – Preprocess the raw data

- Add a second timeline named `delta_t`, such that the head of each DataFrame looks like

	ax	ay	az	|a|	delta_t
time					
2020-09-10 14:08:22.287	-0.66	0.64	1.75	1.97	00:00:00
2020-09-10 14:08:22.297	-0.61	0.40	1.18	1.38	00:00:00.010000
2020-09-10 14:08:22.307	-0.47	0.12	0.51	0.70	00:00:00.020000
2020-09-10 14:08:22.317	-0.40	0.01	0.01	0.40	00:00:00.030000
2020-09-10 14:08:22.327	-0.31	-0.05	-0.48	0.57	00:00:00.040000

- Remove the transition periods at the beginning and the end of the recorded signals (eyeballing is acceptable).
- Visualize the signals after pruning the data.
- What is the length of the remaining signals in units of seconds?

Task 5 – Extract

- Create an additional column named `window_idx`, which maps each row to a distinct period of 100 samples. Each window should be identified by a letter, which indicates the activity, and a running number. An example DataFrame should look like:

<table>
<tr><th></th><th>ax</th><th>ay</th><th>az</th><th>|a|</th><th>delta_t</th><th>window_idx</th></tr>
<tr><th>time</th><th></th><th></th><th></th><th></th><th></th><th></th></tr>
<tr><td>2020-09-10 14:08:26.305</td><td>-1.69</td><td>1.40</td><td>20.87</td><td>20.98</td><td>00:00:04.018000</td><td>w00</td></tr>
<tr><td>2020-09-10 14:08:26.313</td><td>-3.45</td><td>-1.89</td><td>13.07</td><td>13.64</td><td>00:00:04.026000</td><td>w00</td></tr>
<tr><td>2020-09-10 14:08:26.323</td><td>-6.37</td><td>-4.00</td><td>8.09</td><td>11.04</td><td>00:00:04.036000</td><td>w00</td></tr>
<tr><td>2020-09-10 14:08:26.333</td><td>-7.58</td><td>-5.94</td><td>0.61</td><td>9.64</td><td>00:00:04.046000</td><td>w00</td></tr>
<tr><td>2020-09-10 14:08:26.344</td><td>-6.50</td><td>-5.84</td><td>-3.74</td><td>9.50</td><td>00:00:04.057000</td><td>w00</td></tr>
</table>

- Concatenate the DataFrames from both activities to one large DataFrame. Document the shapes of the DataFrames before and after the concatenation.
- How many unique values of column `window_idx` have you generated?
- Use `tsfresh.feature_extraction.extract_features` in order to characterize the activity windows.
- What are the dimensions of the resulting feature matrix? How many features have been extracted?

Task 6 – Activity Recognition Model

- Reduce the number of features by using `tsfresh.transformers.FeatureSelector`. Which are the five features with smallest p-values?
- Visualize these five features using `seaborn.pairplot`. What do you observe?
- Evaluate the performance of a Random Forrest Classifier for recognising the two activities.
  - Use a 10-times repeated 10-fold cross-validation for this purpose.
  - The classifier should only use the statistically significant features as identified by tsfresh.

Submit your report

- Export your Jupyter notebook as html.
- Upload your report to canvas by Wednesday 7th October 10:00pm.