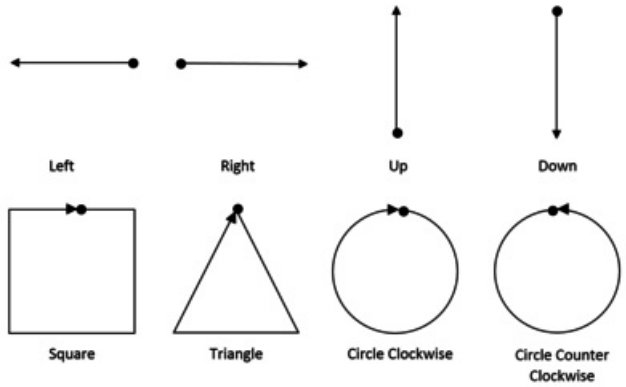
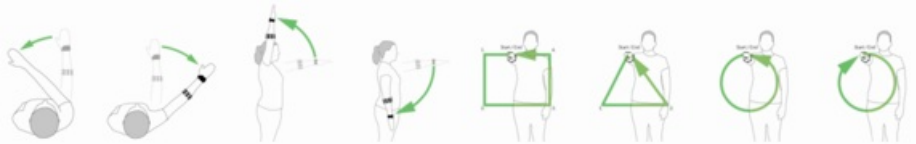
Gesture Recognition

Dataset

- The dataset contains different arm gestures from different participants.
- Recording device and wearing position: smartwatch (LG Watch G) running Android Wear, worn at right wrist.
- Recorded data: 3 axes acceleration measured in G [9.81m/s^2], with a target sampling rate of 50Hz.
- Gestures, participants and samples: 8 gestures, 9 participants. Each participant performed each gesture 30 times, which results in a total of 240 samples per participants, 270 samples per gesture and 2160 samples in total in the data set.
- Acceleration recordings are split per axis into separate csv files (one file per axis). This implies that e.g. the Nth line of each file belong to the same gesture, participant and sample and represent the according x, y and z axis acceleration recordings.
- Csv file columns in this order:
  1. gesture
  2. participantNr
  3. sampleNr
  4. N acceleration values (different lengths per sample)

Gestures in the dataset





Task

- Goal is to build (a) successful gesture recognition model(s). The model should learn to distinguish between N gesture acceleration recording, then be able to decide for new recordings which gesture it shows. Load data, preprocess it, and derive features (see hints below).
  - Do a gallery dependent data partitioning and train models using repeated cross validation. Show you results as at least confusion matrices for the complete train and test partition over gestures. What is the best feature derivation/model configuration you can come up with? Are there gestures that are harder to distinguish than others?
  - **Bonus objective**: do gallery independent data partitioning. As there is a limited amount of participants: leave out the additional, initial split into train and test partition, but do gallery independent cross validation. Thereby, each partition contains data of one participant (e.g. by setting the `trainControl(index=...)` parameters). Show your results at least as some metric (e.g. Kappa) over participants and gestures (e.g. confusion matrix of gestures with mean recognition rates or boxplot of gesture recognition performance with spread over different people). What is the best feature derivation/model configuration you can come up with? Are there any participants and gestures that are harder to distinguish than others?

Hints

- Use vectorized commands and clever parametrization of read.table to load all csv files at once - similar to:

`d <- lapply(FILES, read.table, sep=',', fill = T, col.names = c('gesture', 'person', 'sample', paste('acc', 1:1000, sep='')))`.

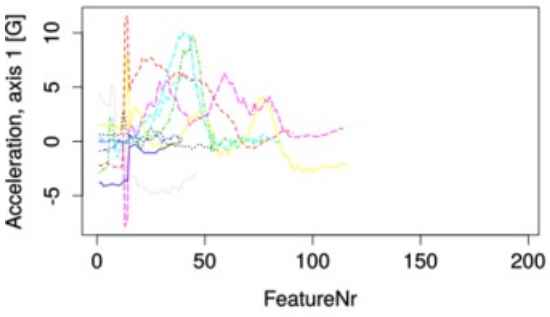
- The `matplot` command can be helpful in plotting multiple timeseries at once, e.g. for comparison. Example call: `matplot(t(myTimeSeriesDataframe[1:10,indexes_of_features]), type='l', col=factor(myTimeSeriesDataframe[1:10,index_of_class]))`.
  Be aware that regular dataframes needs to be transposed using `t(...)` for usage in `matplot`. With using `library(scales)` and the `matplot` parameter `col = alpha(colour = 1, 0.05)` you can make plotted lines transparent, which can be useful when visualizing many samples at once.
- Recorded acceleration should probably be filtered to reduce noise. For example, `?stats::runmed` can be useful here. Hint: you probably need to apply a stronger filter than would seem intuitive at the beginning. Think about *which* noise should be filtered for this task and check if you still have required information in the data after filtering (e.g. using plots). Again, try to use vectorized commands.
- If you plan on using acceleration values themselves as features, different length of recording is going to be problematic. You can interpolate series to uniform length using e.g. `stats::approx`. Again, try to use vectorized commands. Be aware that with interpolation to a uniform amount of features you also lose the implicit information of how long the recording was (which was expressed by the amount of samples before). The length of the original recording can therefore become a numeric feature itself. Hint: you probably need less interpolated acceleration values than you would expect. Usually, human body motion can produce up to a max of about 20Hz of movement frequencies, so using e.g. 100Hz recordings does not necessarily add any information you can use. And the important acceleration information to distinguish gestures is likely well below 20Hz, so a couple of values per gestures probably already do the job.
- Either acceleration values of all axes can be used directly, or their acceleration magnitude can be computed. Latter discards acceleration orientation information. Think about if orientation is important for your task, then use features accordingly.
- Feature derivation is not limited to those discussed in the lecture. But for a start:
  - Frequency power and phase (`Mod(fft(...))` and `Arg(fft(...))`).
  - Autocorrelation of acceleration values (`stats::acf(...)`).
  - Wavelet components (e.g. `library(wmtsa)` with `?wavDaubechies` and `?wavDWT`).
  - All of the above can be applied on the integrations of acceleration as well (position <--> speed <--> acceleration).
  - **Be aware that a) filtering of original acceleration values (noise reduction) will impact your derived features and b) these features should be derived for individual axes or magnitudes, but not a single vector of values of all axes concatenated together. Concatenating features to a feature vector should be done after features have been derived for individual axis or magnitudes.**

Hand-in

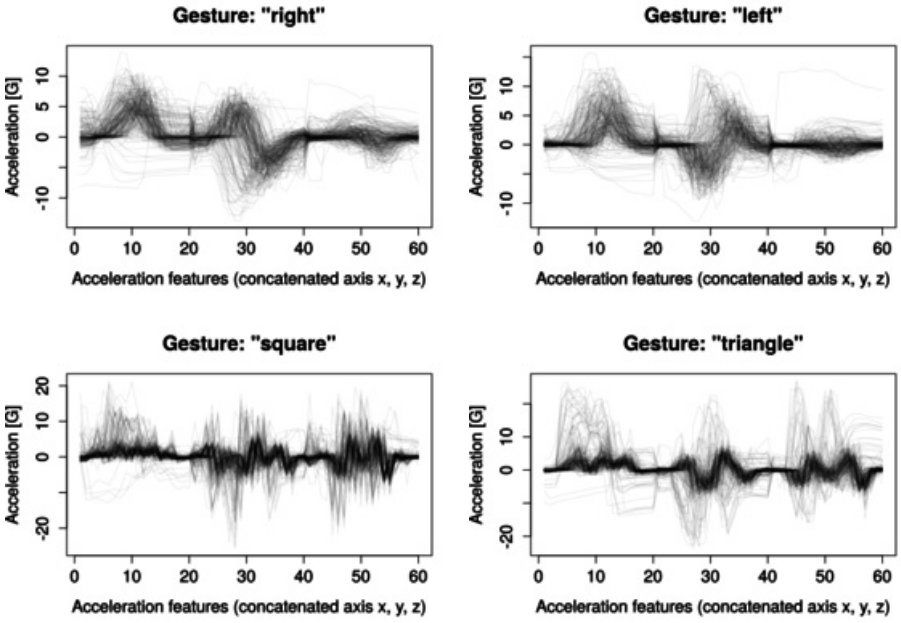
- You can chose tools as you like and can span your toolchain across multiple platforms/languages. For R and alike parts hand in: source code + exported model object stored with `saveRDS` as `RData` file; for Weka parts hand in: detailed description of steps done and exported model. If models are too big for uploading, simply skip them in your export.
- In the report: document everything so that **your system could be rebuilt from using the report only**:
  - Flowchart/block diagram of the toolchain you use, containing each step from the original data to model training and evaluation performance testing.
  - State clearly how you split data/do data partitioning.
  - State clearly which feature derivation/transformations with which parameters you apply.
  - State and compare performance of different models/approaches.
  - State which model you would chose for a real world application, why you would do so, and its performance (confusion matrix, ...).
  - Don't forget to state your thoughts/findings on your data at each point in the toolchain (why you apply transformations, what you see, etc).
- This assignment will be credited by how creative you were, how precise you performed and documented the task, and of course by your findings and solutions. Therefore, the report is highly important for this assignment: everything you want reviewers to *understand* needs to be stated in it.

Examples

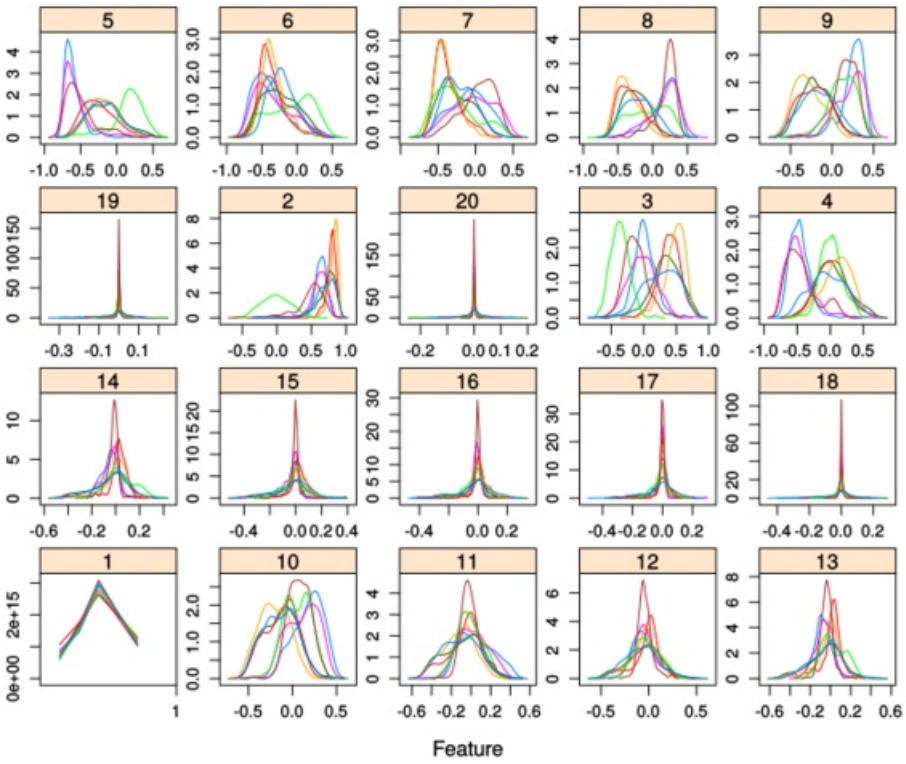
Visualizing your data/results in a similar way will be beneficial. Gesture samples:



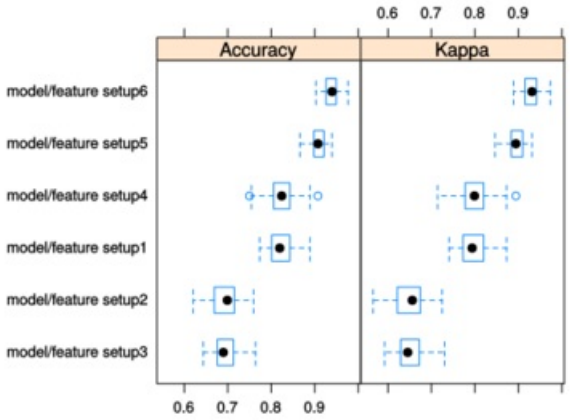
Filtered and interpolated gesture samples:



Example features (excerpt):



Possible model performances:



Possible confusion matrix: