Mobile Sensing: Spring 2015
Exercise: 4

Due on 9th April 2015 by 17:45 PM.

Instructions: All course participants are requested to submit their exercise solutions electronically to the instructors (samuli.hemminki at cs.helsinki.fi and teemu.pulkkinen at cs.helsinki.fi), as well as to the course lecturer (petteri.nurmi at cs.helsinki.fi) by the due date (latest before the exercise session). In all the exercises, do not just give the answer, but also the derivation how you obtained it. Participants are encouraged to write computer programs to derive solutions to some of the given problems.

Ex 1. Study the article Activity Recognition from User-Annotated Acceleration Data by Bao and Intille.

a) Write a short summary of the paper's *Methodology* (Sec.3) in up to half a page.

b) Which is the most efficient classification method? What are the reasons for this according to the paper?

c) Based on the confusion matrix, calculate precision, recall and f-score for each class. What are the easiest and hardest activity types for the proposed method?

The confusion matrix can be (relatively easily) copied directly from the PDF. The matrix and its labels are provided here for easy access: confmat.csv confmat_labels.csv. Once the matrices have been loaded, things are quite simple. Example code in MATLAB:

```
cfmat = csvread('confmat.csv');
%The 'true positives' are all in the diagonal, i.e. this is what goes in the numerator
correct = diag(cfmat);
%Then we just need to divide the true positives with the respective sums, either by rows
precisions = correct./sum(cfmat)';
%or by columns
recalls = correct./sum(cfmat,2);
%F1-score calculation uses formula from lecture slides (with some vectorization added)
f1s = 2*(precisions.*recalls)./(precisions+recalls);

%Output results:
[precisions,recalls,f1s]

ans =

    0.8177    0.8971    0.8556
    0.8686    0.8210    0.8441
    0.8933    0.9478    0.9197
    0.9254    0.9749    0.9495
    0.8815    0.9567    0.9176
    0.8073    0.8867    0.8451
    0.7786    0.7748    0.7767
    0.9285    0.9179    0.9232
    0.9178    0.8768    0.8968
    0.8830    0.9629    0.9212
    0.7611    0.4142    0.5365
    0.8681    0.8251    0.8460
    0.7547    0.8109    0.7818
    0.9086    0.9641    0.9355
    0.7733    0.9514    0.8531
    0.9706    0.9496    0.9600
    0.8177    0.8527    0.8349
    0.6869    0.8561    0.7622
    0.8353    0.4358    0.5727
    0.3290    0.7056    0.4488

%Then we need to look at the easy/hard activities:
%Read the labels
f = fopen('confmat_labels.csv');
%Parse them into cells
labels = textscan(f,'%s','delimiter','\n');
%Resulting cell-structure is a bit awkward, we actually just need the first index
labels = labels{1};

%Sort the f1-scores, and grab just the indices
[~,easiest] = sort(f1s,'descend')
%Then we just use these indices to output the labels (from easiest to hardest):
labels(easiest)

ans =
    'lying down'
    'computer work'
    'vacuuming'
    'reading'
    'bicycling'
    'sitting relaxed'
    'standing still'
    'running'
    'walking'
    'folding laundry'
    'strength train'
    'eating/drinking'
    'walking/carry'
    'brushing teeth'
    'scrubbing'
    'watching TV'
    'climbing stairs'
    'riding elevator'
    'stretching'
    'riding escalator'

%This matches quite closely with what the authors suggested
```

Ex 2. Load accelerometer data from a set of activities: Biking,Cleaning, Climbing, Driving, Running, Walking, Sitting/Working. Implement the method presented in Bao and Intille work in three parts:

a) Preprocess the data by framing the measurements to frames of length 6.7 seconds and 50% overlap.

b) Extracts features: *Mean, Energy* and *Frequency-domain entropy* from each frame.

c) Classify the frames using an optional classification technique (e.g.,Decision Tree, Naive Bayes or k-means). Evaluate the results using 75% of each activity for classifier training, and 25% for testing. Calculate precision, recall and f-score for each class.

   Matlab: *fitctree, NaiveBayes, kmeans*, Java: *Weka*, Python: *SciKit*

   One implementation:
   classifyActivities, which uses createFeatures.