Activity recognition and energy expenditure estimation

A practical approach with Python

WebValley 2015
Bojan Milosevic
Scope

• Goal:
  – Use wearable sensors to estimate energy expenditure during everyday activities

• Specifications:
  – Unobtrusive (minimum number of sensors and devices)
  – Real time (low computational cost)
  – Low power consumption
Outline

• Energy Expenditure (EE) estimation
• Activity recognition
  – Processing pipeline
• Python tools for data analysis
  – Examples and exercises
Energy Expenditure Estimation

• Monitoring of EE is an important step in tracking personal activity and preventing chronic diseases
• Different commercial solutions (FitBit, Nike+, Active, …)
Activity Recognition

- The EE estimation is based on the activity performed by the user
- Wearable systems allow continuous and unobtrusive monitoring
Activity Recognition

- Inertial sensors (accelerometer, gyroscope) are used to track motion
- Wearable sensors (e.g. wristbands) or sensors embedded in commonly used devices (e.g. smartphones)
- Sensor data can be processed in real-time or logged for offline analysis and evaluation
Activity Recognition

- Raw sensor data is pre-processed to eliminate noise and adapt for the next stages
  
- Examples: *calibration*, *low/high pass filtering*, *normalization*, *offset compensation*
Activity Recognition

- Feature extraction is the transformation of the input data into a reduced set of features that extract the relevant information.

- For time series it is usually applied to a window of acquired data.

- Examples: \textit{mean}, \textit{variance}, \textit{magnitude}, \textit{max range}, \textit{mean crossing rate}, \textit{spectral density}, \textit{FFT coefficients}, ...
Activity Recognition

• Application of a classification algorithm to recognize the performed activity

• Use of supervised classifiers, trained with a collected dataset

• Examples: \textit{kNN}, \textit{Trees}, \textit{SVM}, ...
Activity recognition dataset

- Human Activity Recognition Using Smartphones Data Set (HAR Dataset)
- Up to 30 subjects
- 6 activities (WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING)
- Recorded wearing a smartphone (Samsung Galaxy S II) on the waist
- 3-axial linear acceleration at 50Hz
- Pre-segmented in separate files:
  - One file for subject and activity (e.g Subject_2_LAYING.txt)
  - Each row of the files contains the X, Y and Z accelerations
Offline data analysis

• We will perform the needed processing pipeline offline
• Used to test and compare different algorithms and parameters, obtaining performance and computational evaluations
• The optimized algorithm can be easily implemented on the final platform for online (real-time) use

• Steps:
  – Read and plot data
  – Organize and pre-process data
  – Compute features
  – Train classifier and test accuracy
Python for data analysis

- Python is a powerful **high level scripting language**
  - Open source
  - Works on all platforms
  - Wide community of developers and users
  - Many libraries to support specific functions
- Libraries for data processing and visualization
  - Pandas, numpy, matplotlib, scikit, ...
NumPy

• NumPy is the fundamental package for scientific computing with Python.
  – a powerful N-dimensional array object
  – sophisticated (broadcasting) functions
  – tools for integrating C/C++ and Fortran code
  – useful linear algebra, Fourier transform, and random number capabilities

• Tutorial: http://wiki.scipy.org/Tentative_NumPy_Tutorial
Import data from file

• **Python offers standard file I/O functionalities**
  ```python
  # read from file
  in_file = open("test.txt","r")
  text = in_file.read()
  in_file.close()
  ```

• **To read data files** is more useful `loadtxt(filename)` from `numpy`
  – fast reader for simply formatted files
  – each row must have the same number of values

• **Example:**
  ```python
  import numpy as np
  filename = './data_sample/Subject_2_LAYING.txt'
  # load data from text file
  data = np.loadtxt(filename)
  ```
Plot data

- *matplotlib* is a useful library to easily **create and customize plots**
- Tutorials: [http://matplotlib.org/users/pyplot_tutorial.html](http://matplotlib.org/users/pyplot_tutorial.html)  
- Example:
  ```python
  import matplotlib.pyplot as plt
  # create figure and plot variable data
  plt.figure()
  plt.plot(data)
  # show created plot(s)
  plt.show()
  ```
- `show()` is a **blocking function** and should be called only once

⇒ *script_data_plot.py*
Plot data

• Look at the documentation and examples to **customize your plot**, change the proprieties and add information...

```python
# create figure and plot data
plt.figure()
plt.plot(Time, data, linewidth=2.0)
# Set plot proprieties
plt.xlabel('Time [s]')
plt.ylabel('Acceleration [g]')
plt.title('Laying')
plt.grid(True)
```

⇒ *script_data_plot_extended.py*
Import dataset

• Activity recognition dataset is organized in separate files

• Different ways to import and organize the data:
  – We can read the data in separate variables or lists (e.g. divided by Subject or action)
  – We can read and aggregate the data in a combined variable (e.g. all actions by a Subject)
  – Use lists, numpy arrays and the functions to vertically and horizontally stack arrays and matrices

⇒ script_import_data.py
⇒ script_import_data_combined.py
Process data

• Apply algorithms to process the acquired data
• Always check if the desired function is already implemented in the library:
  – numpy.linalg, numpy.fft, statistic functions, ...
• Example: compute the norm of each acquired acceleration vector (use np.linalg.norm)

  for t in range(len(data)):
    norm[t] = np.linalg.norm(data[t])

⇒ script_norm.py
Compute features

- Use **sliding window** approach to compute features and extract information from the acquired data.
- Two consecutive windows can have an overlap and share data, depending on the increment between the two windows (STEP).

Parameters: WINDOW_SIZE, STEP are application dependent and should be tested.
Compute features

• Use **sliding window** approach and compute **mean** and **variance** of the sampled accelerometer signals
• Use functions provided by the library

```python
for i in range(col):
    for j in range(0, row, STEP):
        # j = index of first element in current window
        # j+WINDOW_SIZE is the window of current elements
        # compute mean of window elements
        data_mean[j:j+WINDOW_SIZE-1,i] = np.mean(data[j:j+WINDOW_SIZE-1, i])
```

⇒ *script_mean.py*
Compute features

• Compute features for all the files in the data_sample folder and arrange them
  – Create separate variables for the features and do not repeat instances of features
  – Aggregate all the features in one variable (matrix with one column for each feature)
  – Create labels for each activity (e.g. LAYING = 1, SITTING = 2, ... )
  – Save features in files so you can use them easily later

⇒ script_features.py
Train classifier

• Use the computed features to train and test a classifier to recognize the different activities.
• Library *scikit* has functions for main machine learning algorithms
• Prepare the features and separate them for each activity
• Separate the features in a training set and a test set
  – Take 20% for training, the rest for testing
  – Try to take randomly distributed features for training

```python
from sklearn import tree
clf = tree.DecisionTreeClassifier()
# Train
clf = clf.fit(dataTrain, labelsTrain)
# Predict
labelsPredict = clf.predict(dataTest)
# Prediction accuracy
score = clf.score(dataTest, labelsTest)
```
Collect your dataset!

• Use a data logging app on your smartphone to collect data during various activities.
  – Several free applications available (AndroSensor for Android, SensorLog for iOS)
• You can log your activities throughout the day and annotate what you were doing.
• Share and compare the acquired data with your friends
• Process and classify the data as we have seen for the HAR Dataset
Exercises

• Import the HAR dataset for different Subjects and organize it in lists and in an aggregated variable.
• Acquire your own dataset, import and plot it.
• Compute different features: mean, variance, peak-to-peak difference, range, mean crossing rate, energy, FFT coefficients, ... (look for useful features and compute them both on the single axes and on the norm)
• Test the classification results with different features and different WINDOW_SIZE and STEP parameters
• Try different classifiers (tree, kNN, SVM, ...)
• From the classified actions estimate the EE using the tables with the metabolic rates and the physical characteristics