Human Activity Recognition using mobile sensors

Innopolis University

Machine Learning Fall 2019 -Bachelors-

Human activity recognition is the problem of classifying sequences of accelerometer data recorded by specialized sensors or smart phones.

Automatic recognition of physical activities – commonly referred to as human activity recognition (HAR) – has emerged as a key research area in human-computer interaction (HCI) and mobile and ubiquitous computing. One goal of activity recognition is to provide information on a user's behavior that allows computing systems to pro-actively assist users with their tasks.

HAR's applications cover a variety of real world problems in fields like sports, medicine, authentication, localization and Industry Manufacturing and Assisting.

Taking the medical field for example one of the many applications of HAR is Elderly Care, where we find a growing need in (both physically and mentally), partially because of the retirement of the baby Boomer generation. A major goal of the current research in human activity monitoring is to develop new technologies and applications for elderly care Those applications could help prevent harm, e.g., to detect older people's dangerous situations.

The dataset we will be using is available on the UCI Machine Learning Repository for free datasets.

https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones

The experiments have been carried out with a group of 30 volunteers within an age bracket of 19-48 years. Each person performed six activities (WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING) wearing a smart phone (Sam-sung Galaxy S II) on the waist. Using its embedded accelerometer and gyroscope, we captured 3-axial linear acceleration and 3-axial angular velocity at a **constant rate of 50 Hz**. The experiments have been video-recorded to label the data manually. The obtained dataset has been randomly partitioned into two sets, where 70% of the volunteers was selected for generating the training data and 30% the test data.

Folder Structure:

```
.
├── train                               # Training Data
│   ├── Inertial Signals                # RAW DATA
│   │   ├──acc-[x,y,z][x,y]train.txt    # body accelorometer
│   │   ├──gyro-[x,y,z][x,y]train.txt   # gyroscope
│   │   └──total-[x,y,z][x,y]train.txt  # total accelometer
│   ├── Subject_train.txt               # person performing activity
│   ├── X_train.txt                     # pre-processed X
│   └── y_train.txt                     # labels
├── test                                # Testing
```

```
│   ├── Inertial Signals                # RAW DATA
│   │   ├──acc-[x,y,z][x,y]test.txt     # body accelorometer
│   │   ├──gyro-[x,y,z][x,y]test.txt    # gyroscope
│   │   └──total-[x,y,z][x,y]test.txt   # total accelometer
│   ├── Subject_test.txt                # person performing activity
│   ├── X_test.txt                      # pre-processed X
│   └── y_test.txt                      # labels
├── Activity_labels.txt                 # labels and their id
├── features.txt                        # feature names
├── features_info.txt                   # feature's description
└── README.md                           # README
```

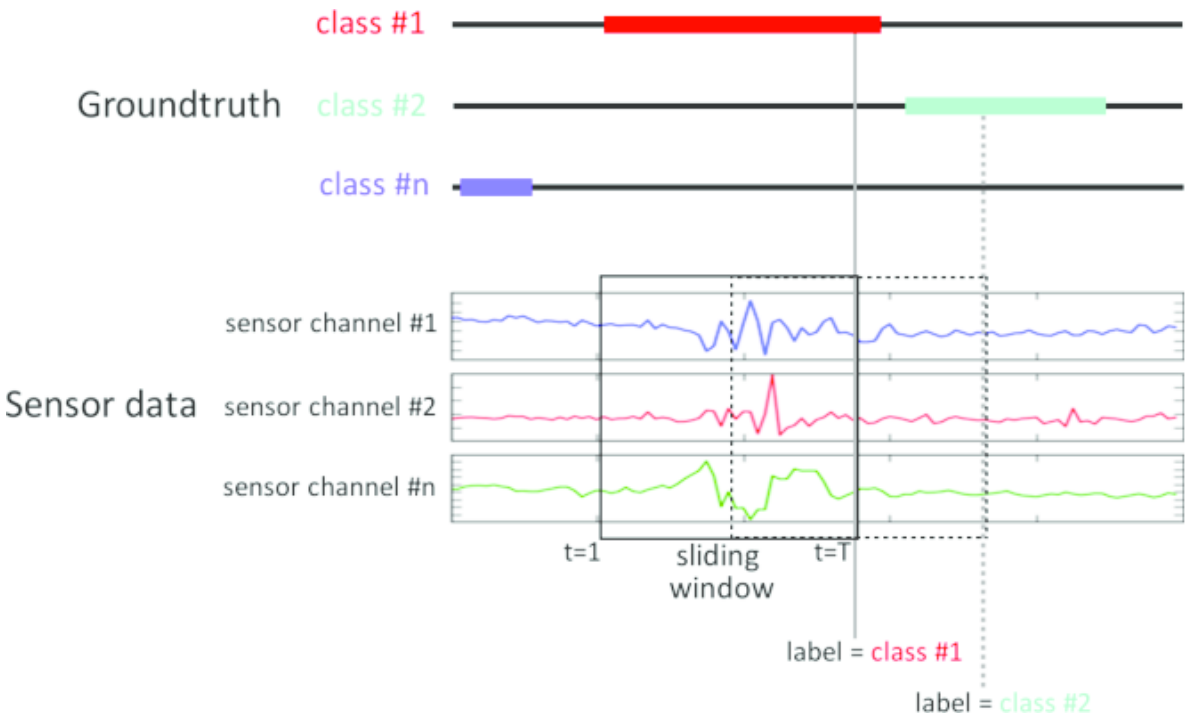
In this Homework you will implement a variety of linear classifiers and compare their performances and training time.

The homework is divided into four sections:

1. Data Reading and preprocessing.
2. Model Creation.
3. Hyper-parameter Tuning (using cross-validation).
4. Comparison between models.

1. Data Reading and preprocessing

The sensor signals (accelerometer and gyroscope) were pre-processed by applying noise filters and then sampled in fixed-width sliding windows of **2.56** sec and **50%** overlap (**128 readings/window**). The sensor acceleration signal, which has gravitational and body motion components, was separated using a Butterworth low-pass filter into body acceleration and gravity. The gravitational force is assumed to have only low frequency components, therefore a filter with 0.3 Hz cutoff frequency was used. From each window, a vector of features was obtained by calculating variables from the time and frequency domain.



Task

1. Load Dataset 1:
    1. *HAR Dataset/train/X_train.txt*
    2. *HAR Dataset/train/y_train.txt*
    3. *HAR Dataset/test/X_test.txt*
    4. *HAR Dataset/test/y_test.txt*

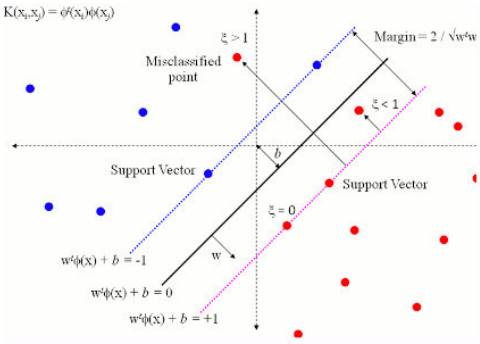
PS: the data set has already been pre-processed.

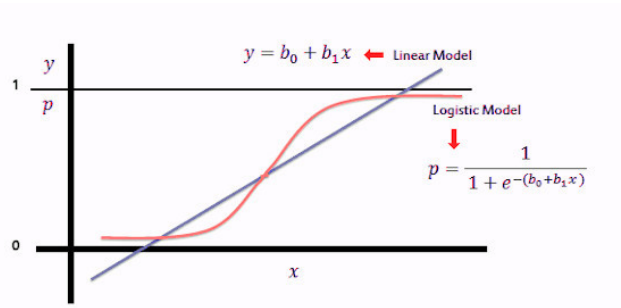
2. Model Creation

Now we will define a list of linear models:

1. Logistic Regression
2. SVM (with two variants)
3. SGDClassifier

Use the define_models() function to build the list of all models and their respective potential hyper-parameters.

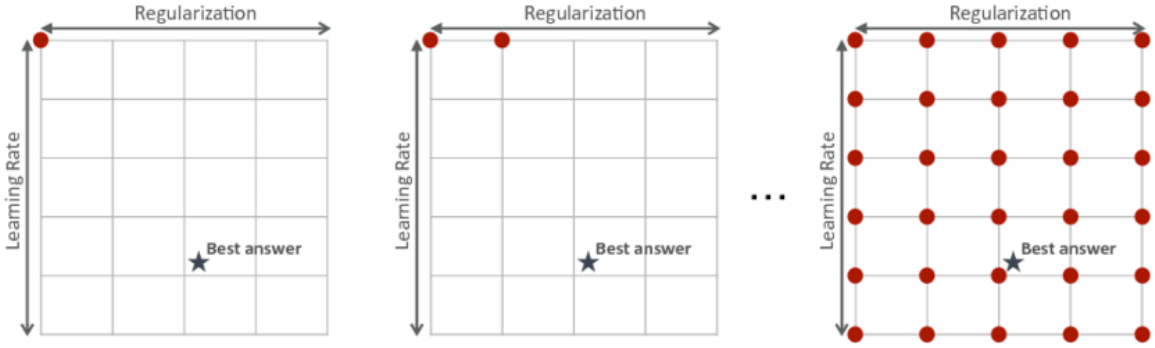




3. Hyper-parameter Tuning (using cross-validation).

To choose the best hyper-parameters for each model we will use Grid-Search-CV.

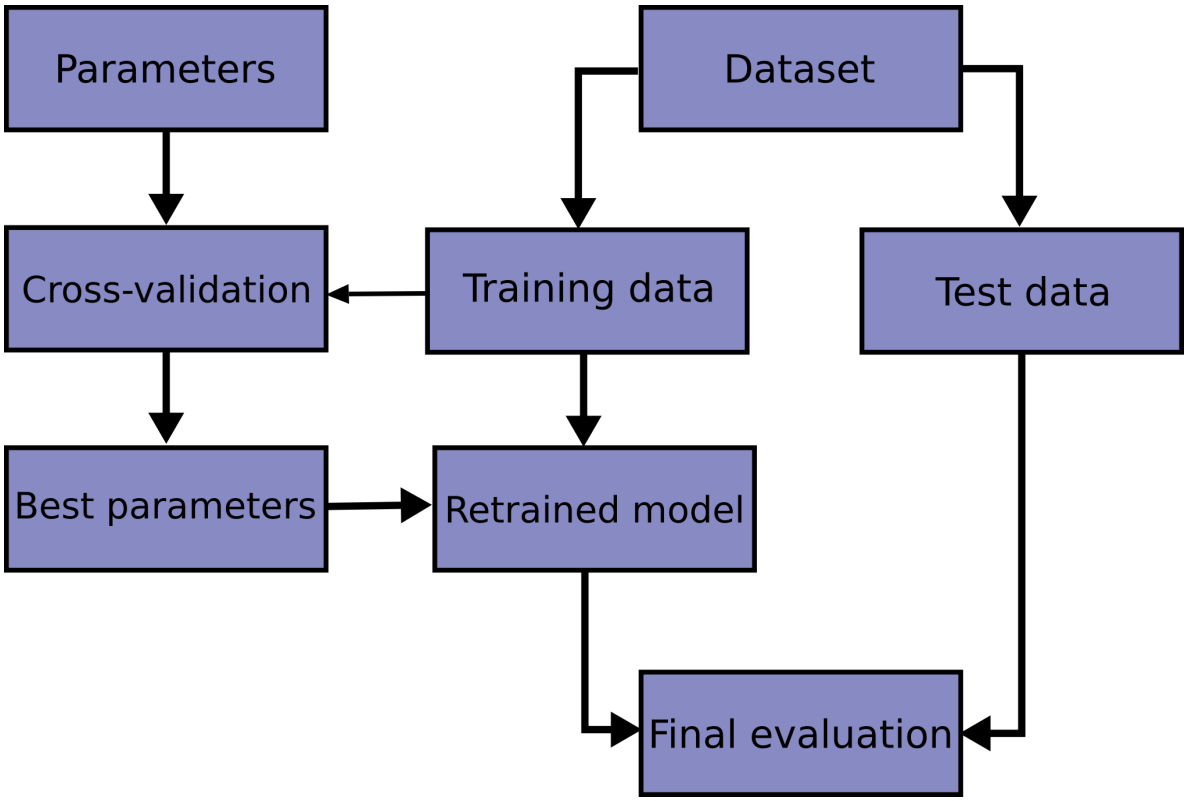
Grid-Search: Is an exhaustive hyper-parameter tuning algorithm that checks all the possible hyper-parameter combinations and chooses the best one of them. For evaluation of every set of hyper-parameters, Grid-Search-CV uses cross-validation to evaluate the model's performance.



4. Comparison between models

After we found the best set of parameters for each model we will evaluate them on the separate test data-set using:

1. Accuracy, Precision, Recall.
2. Confusion Matrix.



Submission

Please submit the following:

1. homework.py file:
    1. You can use lab code for reference.
    2. Cheating IS NOT ACCEPTABLE.
    3. Make sure that all the tests inside the file pass before submitting.
    4. please DO NOT use .ipynb. (violation of this requirement leads to -10% from the grade).
    5. Changes to the structure of the functions will result in -25% from the grade.
2. report.pdf
    1. description of the results.
    2. which model you think is better.
    3. which model takes more time.
    4. the feasibility of precision and recall in this case.