

HAR Document

Revision data: 29th, September, 2016, Revised by Huoyuan Tan

HAR, short for Human Activity Recognition. This project was established by Xiaomu Luo. The source code on GitHub is:

<https://github.com/woodwood2000/HAR/tree/dev>

This document is a brief introduction to this project. We will not have detailed introduction about the algorithm and the project background but you can learn more about them by reading the articles or thesis listed below:

- (1) Abnormal Activity Detection Using Pyroelectric Infrared Sensors
<http://www.mdpi.com/1424-8220/16/6/822/pdf>
- (2) Human Indoor Localization Based on Ceiling Mounted PIR Sensor Nodes
<http://ieeexplore.ieee.org/document/7444903/>
- (3) Design and implementation of a distributed fall detection system based on wireless sensor networks
<https://www.researchgate.net/publication/257877511>
- (4) Human Tracking Using Ceiling Pyroelectric Infrared Sensors
<https://www.researchgate.net/publication/224113130>

More papers about this project can also be found or downloaded on the website.

This document will be organized as follows. In the first part, we will have an overview of the project construction. We will talk about more details about the project even the function in the second part.

Part 1 An Overview of the Project Construction

A. What we had done in the source code

Up to the revision data, we had fundamentally finished three tasks, training the models, activities recognition and show the recognition results online. There are significant improvements still to be made. We hope you can join us and make this project more perfect. We did our work mostly on the dev branch on GitHub including added some functionalities or made correction.

For activities recognition, we had several experiments in different algorithms, features and time window sizes or different combination of them. Eight kinds of activities recognition in different time window sizes and different algorithms realizes in the main function `prepData.m`. Two kinds of activities recognition like walking vs others in different time window sizes and different algorithms can be found in the main function `classifyActionState.m` and its subfunction of subfolders are also end up there names with `**_ClassifyState` or `**Classify`.

To predict the new received data and show the recognition results online, our

main function is dataProcess_online.m and its subfunctions and subfolders are end up there names with ****_Oline** or **online**.

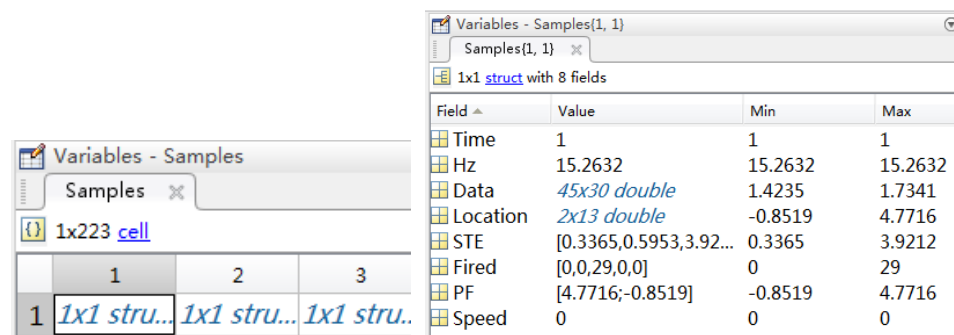
Beyond that, in the subfolders tools there are some functions that can evaluate the effect of the models and be reused in different models.

B. How we structuralize our data received from PIR sensors

The data structure we used refer to the project Tim had done:

<https://sites.google.com/site/tim0306/>.

In our project, all samples are saved as cells, more information such as Time, frequent, voltage, location and speed are saved as structures, which are shown in Fig.1. .



Field	Value	Min	Max
Time	1	1	1
Hz	15.2632	15.2632	15.2632
Data	45x30 double	1.4235	1.7341
Location	2x13 double	-0.8519	4.7716
STE	[0.3365, 0.5953, 3.92...]	0.3365	3.9212
Fired	[0, 0, 29, 0, 0]	0	29
PF	[4.7716; -0.8519]	-0.8519	4.7716
Speed	0	0	0

Fig.1. Structuralized data

Part 2 More Detail About the Project

A. More details about models training and models selection

Although there are eight or two kinds of activities recognition using different algorithms, features and time window sizes in the existing source code, the usage and the structure of them are very similar, so we will just show some details about eight kinds of model training, activities recognition and the rest you can do some comparative reading which is easy to understand if you have already comprehend the former. More ever, in this document, we will not talk about the basic grammar about Matlab and paste all script to explain step by step, just the main process and what the significant function can do. Further more details can be found in the annotation of the script.

In the main function prepData.m for eight kinds of activities recognition, there are five procedures to realize models training and models selection. The procedures are preprocessing the data received from PIR sensors, partitioning the data into fixed time window size, using the training dataset to train classification model and predict the results using the test dataset, revising the recognition activities and evaluating the model effect, respectively.

To preprocess the data received from PIR sensors, there are more detailed annotation in the subfunction pre_pocessing_CS2.m. Until now, we have five sensors node and each sensor node have nine sensors, so what we mainly do in this process

is recognizing where the message come from and then make them into matrix that can be read.

To segment the data into fixed size, you can call SegData.m, which you can set the size of time window, the gap between two windows, an also translate raw data into other features by using optional variables, calculate other information such as location by calling calcLocation.m, PF by invoking PFilter.m.

To train different classification model using training set and predict the results in the test set, you can call TrainTestSplit.m by setting corresponding variables which means different kinds of classification models. Fig.2. will enumerate some classification algorithm used in this project and show how we use this function.

```
outputNB{i,cross_k} = TrainTestSplit('nb', curExp);
```

a. Naive bayes

```
outputHMM{i,cross_k} = TrainTestSplit('hmm', curExp);
```

b. Hidden Markov Model

```
outputRF{i,cross_k} = TrainTestSplit('rf', curExp);
```

c. Random Forest

Fig.2. Classification algorithm used in outline recognition.

In the subfunction TrainTestSplit.m, we will call modelNameTrainSplit.m to train the classifiers and then call the modelNameTestSplit.m to predict the test set using the trained model output by modelNameTrainSplit.m. So if you want to add more classification algorithm in this project, please name your training and test function in the shape of **TrainSplit or **TestSplit, the ** is the name of your classifiers what you want.

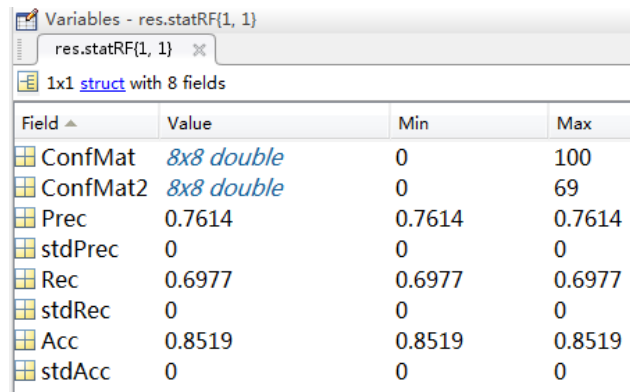
To revise the recognition activities, for the time being, we use CRF (Conditional Random Fields).In this project, by virtue of CRF++ we can easily make CRF come true. More details about CRF++ can be found in its document and we may not talk about it in this document.

In order to evaluate the model effect, we have different assessment methods which are collected in the subfunction calcExtendedResult.m. You can calculate several assessment criteria for different models just call a function conveniently, which are shown in Fig.3. All assessment methods and subfunction calcExtendedResult.m can be found in subfolder tools. The frequently used evaluative criteria such as confusion matrix in percentage or in data, precision, recall, accuracy, ROC, AUC will be calculated and saved in a structure. If you want more detail illustration please read the annotation in the source code.

```
if (exist('outputNB'))
    for i = 1:size(outputNB,1)
        res.statNB{i} = calcExtendedResult(outputNB(i,:));
    end
end
```

```
if (exist('outputHMM'))
    for i = 1:size(outputHMM,1)
        res.statHMM{i} = calcExtendedResult(outputHMM(i,:));
    end
end
```

a. How to call calcExtendedResult for different models.



The screenshot shows the MATLAB Variables window for the variable `res.statRF(1, 1)`. It displays a 1x1 struct with 8 fields. The fields and their values are as follows:

Field	Value	Min	Max
ConfMat	8x8 double	0	100
ConfMat2	8x8 double	0	69
Prec	0.7614	0.7614	0.7614
stdPrec	0	0	0
Rec	0.6977	0.6977	0.6977
stdRec	0	0	0
Acc	0.8519	0.8519	0.8519
stdAcc	0	0	0

b. Several assessment criteria for RF.

Fig.3. Calculate several assessment criteria for different models just call `calcExtendedResult`.

B. More details about data reception

There are two application scenarios in our project, offline and online. For offline scenario, we use a serial port application to receive the sensors data and then save data in the plain text files. When training and selecting the classification model, we read the data from plain text files if needed. More details can be read from the script `prepData.m`.

For online scenario, we use a serial port application written in C# to write the received data to a fixed serial port, then the Matlab mainfunction `dataProcess_online.m` will read the data from previous fixed serial port and deal with them. How the subfunction `pre_processing_online.m` preprocess the received data and how the subfunction `SegData_online.m` segment the matrix data, please read annotation for more details in the source code.

In the same time, in order to simplify the debug process, we don't always open the sensors to receive online data but read data from plain text files and use TCP (Transmission Control Protocol) send data to fixed serial port, which can simulate online sending data. To realize this simulation, you can call script `tcpClient.m`. Before you call this script, a dependent package named `tcp_udp_ip` need to be added to your Matlab path.

C. More details about online show

To use the already trained model predict the new received data and display how well the model perform, in the main script `dataProcess_online.m` we call `rfPredict_Online.m` to predict and `continouseTrajectory_online` to show people trajectory. The predict results are also shown in the same time. The online show will look like Fig.4. and we are looking forward to more visualized results can be added.

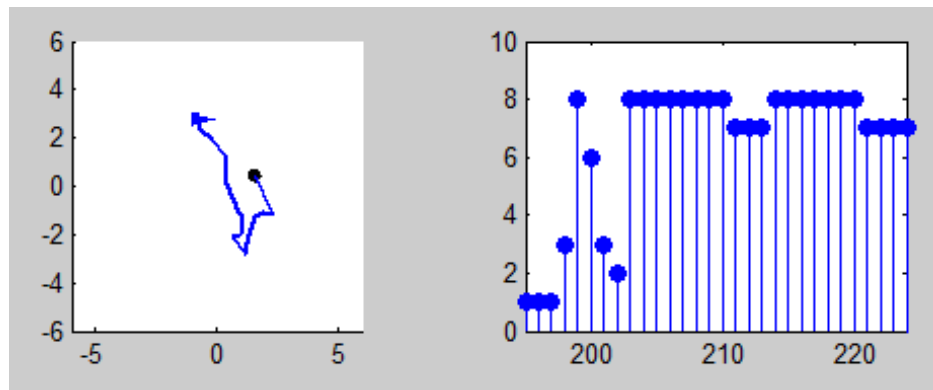


Fig.4. The visualized online results, left for people trajectory and right for predict results.