Lappeenranta University of Technology

BM40A0700 - Pattern Recognition

Practical Assignment

Report - The Jedi Master

2.12.2015

Karri Väänänen - 0358726

Alejandro Antillon - 0444627

Table of contents

1 Introduction

The present report presents the design and implementation of a classifier based on the k-neighbors to classify a set of test data using a relatively larger set of data for training. The project's goal is to successfully classify the movements of a wii remote for a non existent game called "Jedi Master".

The classifier must be able to label the user's movement into one of the possible 7 movements for which it has been trained, these movements are:

1. Slash left
2. Slash right
3. Cross strike
4. Parry & strike
5. Parry up
6. Overhead strike
7. Stab

The device used for data collection is a remote controller with accelerometers in each of the 3 axis, this data although it is already useful, was processed for signal enhancement and more features were extracted from it. Figure 1 shows a set of charts showing the raw input in each axis in the time domain next to its equivalent in the frequency domain, which is one of the features extracted.

The data provided for the assignment was first visualized to see if there are any features exposed to the naked eye. Having this done early was also seen to help during the later stages of the assignment. Both the time domain and frequency domain was plotted.

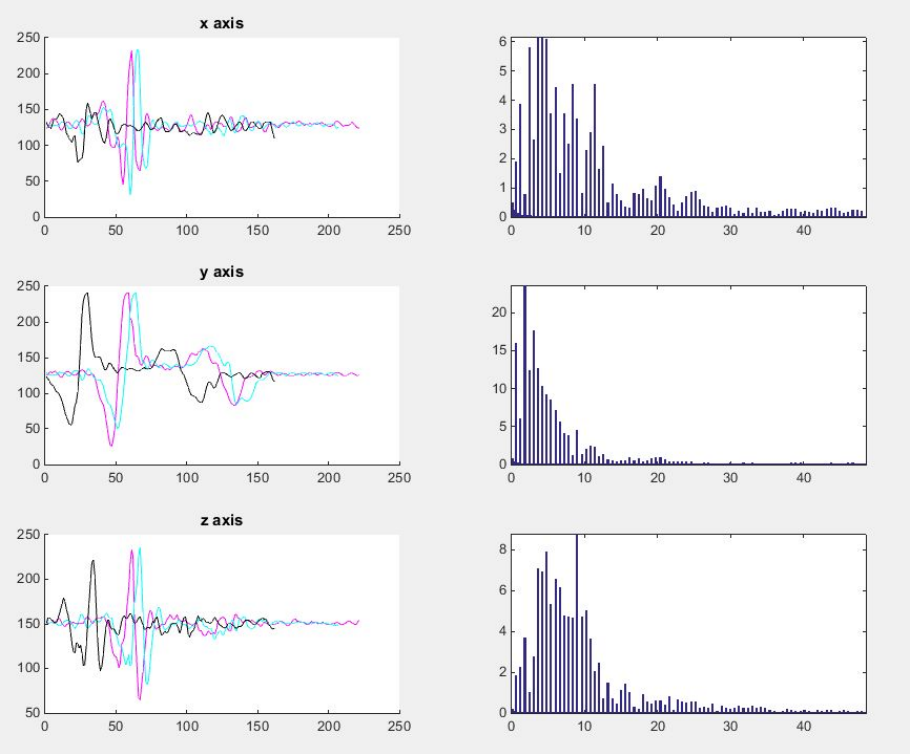

Figure 1: Visualization of accelerometer data from “Stab” movement

2 Selected methods

2.1 Feature extraction

Each axis is considered individually in each feature. Multiple features were extracted from the raw data, then the different features were analyzed and experiments with different combinations were performed to find those that are relevant. The features selected were the following.

2.1.1 Frequency and magnitude

The signals were analyzed not only in the time domain, but also in the frequency domain, the features of interest are the frequency and magnitude of the fundamental frequency of the signal. The computation is performed using the FFT analysis [2].

Before applying the matlab function “fft” signal was purified of the linear trend. Linear trend was noticed to cause distracting peaks at 0 Hz. Since the nature of the data removing this was seen useful in practise. Matlab detrend() function was used to remove linear trend from the signal.

2.1.2 Time value of highest peak

The time value of the highest peak in the acceleration curve, although it is a simple feature that is extracted from the acceleration data, experiments pointed that it was helpful in the classification process.

2.1.2 Velocity, displacement and total distance traveled

The total displacement and total distance traveled in the movement were extracted from the acceleration signal.

The extraction of velocity is obtained by integrating the acceleration signal, and subsequently, displacement is obtained by integrating the velocity signal, and the total distance traveled is the absolute value of the integral of velocity.

Notice that the velocity and displacement data are not subject to any unit, since the units for the acceleration data are not known.

2.1.3 Mean/average, Standard deviation, Minimum and maximum

For the purposes of understanding the signal data several features were extracted. Simple features such as mean, standard deviation, minimum, and maximum were collected as were described in articles [1][3] read during the project.

2.2 Classification

Classifier chosen for the problem was k-nearest-neighbours classifier. K-nearest-neighbour classifier provides an easy way to classify dataset with large amounts of features.

Before the classification normalization of the feature-data was carried out. The differences in the values of features such as displacement and others were considered too large. Without the use of normalization some of the features would have dominated the classification severely. For the normalization the min-max scaling method was utilized. The method was seen appropriate. Note that the same normalization is also applied to test samples.

In our kNN-algorithm, the first four steps are the setup and training of the classifier, this includes loading the training samples and extracting all the features from them, then save the output matrix to a file called “features.mat”, which is later sued by the classifying function, all this setup and training process is encapsulated in a function called “setup” which is included in the deliverables of the project. However the “features.mat” file resulting from the setup process is also distributed with the code, therefore it is not necessary to run the setup function provided that the the features.mat file functions are in the Matlab workspace. The features.mat file can be recompiled if the training set is changed.

The following steps are executed when the “pr_classify” function is called, the function takes the sample to be classified as a parameter and outputs the resulting class, the “pr_classify” function does the following: calculate the Euclidean-distances from the test sample to every training sample. Sorting the distances and selecting the smallest ones gives the classes of the nearest

neighbours. After that the class with highest count is selected. A graphic description of the algorithm is shown in figure 2.

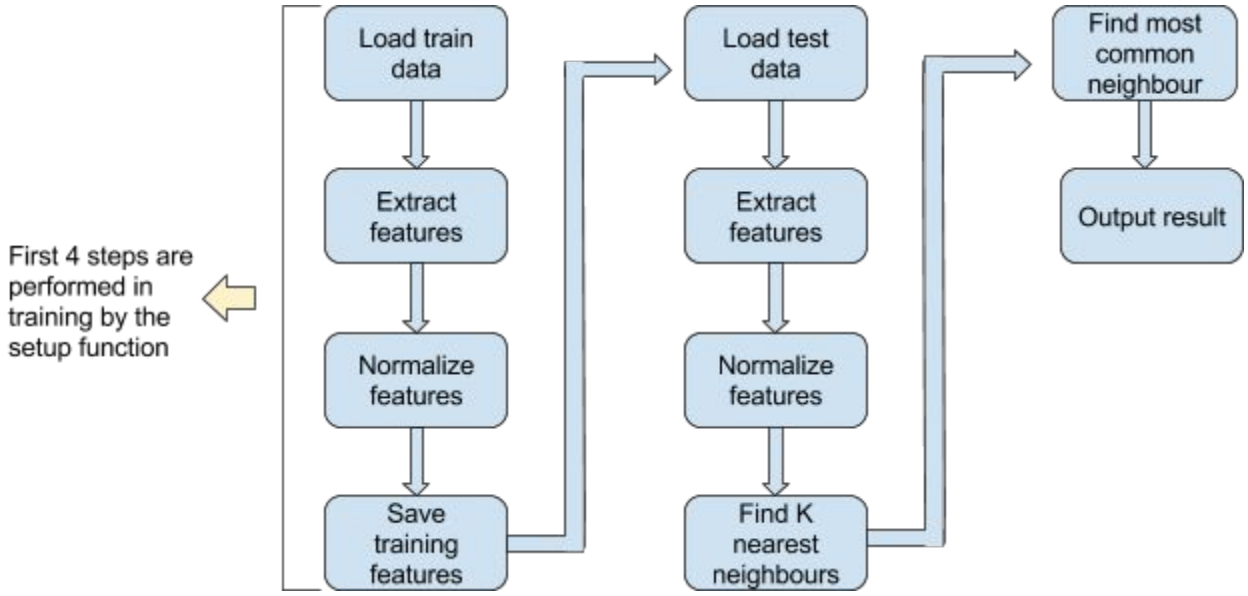

Figure 2: Classification algorithm

3 Results

Leave-one-out testing or leave-p-out testing was conducted to get trustworthy results. Without separating the data to different training and testing sets the classification performance estimation would have been biased producing values such as 100%, and 97% performance for kNN k=1, and k=3 respectively.

Data was divided into testing and training sets randomly. Testing set being only one sample or slightly smaller than the training set. The classification was conducted 1000 times and the average of the results is presented in the table below. Leave-one-out testing, calculated from 1000 subsequent random estimates, gave the final results.

	p=1	p=5	p=10	p=25
classify, default	87%	88%	87%	87%
knn, k=1	95%	95%	95%	94%
knn, k=3	90%	89%	90%	89%
knn, k=5	88%	88%	88%	85%
knn, k=9	73%	73%	74%	72%
p=# of samples in testing set				

Table 1: Leave-p-out testing of classifier

Interestingly, the kNN-classifier seemed to produce the best results using k=1. However, in some cases this may imply overfitting. The kNN-classification with higher k-values may give more robust results without sacrificing performance too much.

References

1. S. J. Preece , J. Y. Goulermas , L. P. J. Kenney and D. Howard "A comparison of feature extraction methods for the classification of dynamic activities from accelerometer data", IEEE Trans. Biomed. Eng., vol. 56, no. 3, pp.871 -879 2009, [Available at http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4663615]
2. Fast-Fourier Transform (FFT), MathWorks Documentation, [Available at http://se.mathworks.com/help/matlab/math/fast-fourier-transform-fft.html]
3. Jeremy W, What is the feature extraction tool and how does it work?, ActiGraph website, Referenced on 1.12.2015, [Available at https://help.theactigraph.com/entries/22119735-What-is-the-feature-extraction-tool-and-how-does-it-work-]