

COMP255 Assignment 1

Human activity recognition project

Introduction

The purpose of this project is to train a machine learning model to recognise different kinds of human physical activity. An activity can be defined by a series of movements made with certain intensity by the human body. Activities can be as simple as flexing an arm or wiggling a toe, but this project is only interested in activities made with the whole human body. These include physical activities commonly associated with exercise such as walking, running, cycling etc. But also included are less strenuous activities such as lying still, sitting down and washing dishes.

This project relies upon which has been sourced from the Daily Life Activities database, originally gathered by Dominik Schuldhaus and Heike Leutheuser. By recording movements in the wrist, chest, hip and ankle they have managed to capture enough hard data to distinguish one activity's movements from another's. With the help of 19 volunteers they have gathered data on 13 activities. The goal of this project will be to train a machine learning model on these gathered datasets and test if it is able to accurately categorise data belonging to one of the 13 activities.

System Design

In order to complete the goal of recognising human activity, the system must first be able to:

1. Convert the datasets into a format readable by the system
2. Extract candidate features from the datasets
3. Train and evaluate a machine model based off the feature sets

Therefore, the system's components should be divided into three different architectural areas: data pre-processing, feature preparation and machine learning.

Data pre-processing contains components which are concerned with the importation of data from different file formats, along with components to visualise and edit the data.

Feature preparation focuses on the creation of a feature training set and requires components to extract features from the data and output them into new training and testing sets.

Machine learning is where the model training takes place. It requires components to train both KNN and SVM algorithms, along with the ability to test a model and output the results of the test.

Key components

Data processing

- Load dataset
Imports data from a static dataset into the system. Will be required to read .csv files and output the data as some form of 2d array. Only required to retrieve data on a per file basis, hence is not required to read specific entries from a file.
- Visualise data
Plots a line graph of selected columns from a data set. Inputs are selected by specifying the participant (0 – 18), the activity (label at the end of each column numbered 1 - 13) and the sensor, which are in groups of 3. Required for the analysis of data sets
- Remove signal noise
Filters a dataset by removing outlying values. Noisy data has the potential to create feature sets which are not truly representative of the data and can therefore train inaccurate classifiers. Required to edit a select dataset by removing entries containing outlying data.

Feature preparation

- Extract features
Converts a raw dataset into a feature set, after processing the dataset as a time series. The dataset(s) represent data recorded from an activity over a period. Therefore, it is required that the data is first divided into segments, based off a

specified time window before features can be extracted from it. Features are properties of all data found within a specific column. Candidate features include the minimum, maximum and average of all a sensor's recordings inside a time window. Each activity has been discreetly labelled so performing overlap segmentation, where segments share data points, should be unnecessary.

- **Prepare training set**
Divides a feature set into training and testing sets. Will be required to specify how the data will be split; what ratio will be used for training/ what will be used for testing. For proper distribution, will be required to set aside testing data from each activity performed by a participant. Will output two datasets.

Machine learning

- **Train models**
Creates machine learning model based from training feature set. Will require the ability to train both KNN and SVM Classifiers.
- **Test models**
Evaluate models using test set and the output feature set from the machine learning training. Requires ability to output classification accuracy score and plot confusion matrix.

Sprint Review

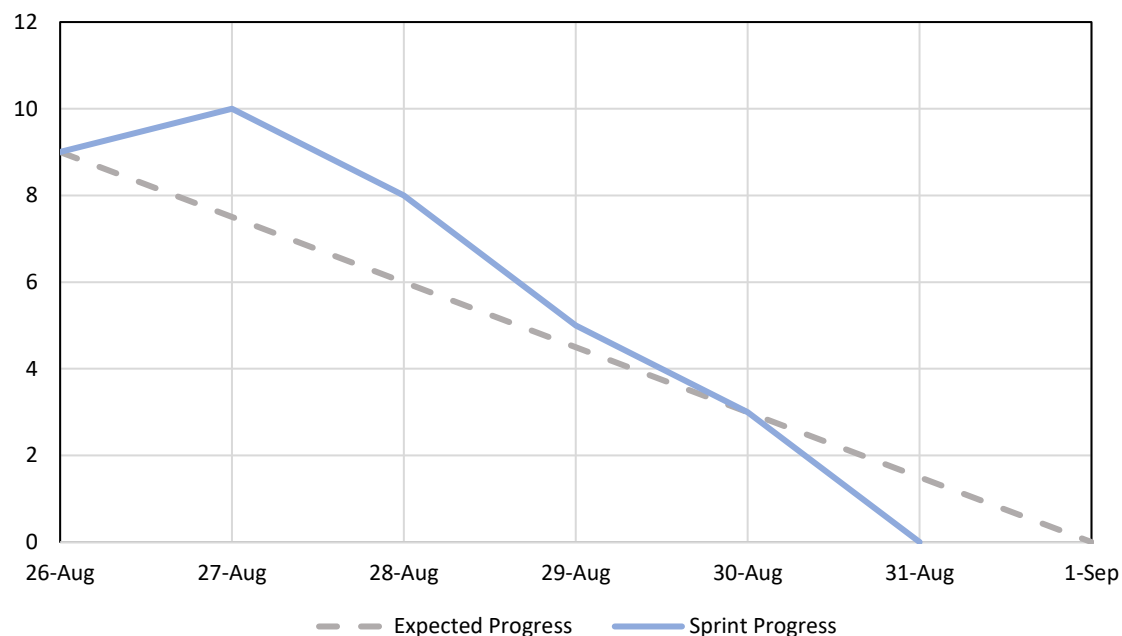
Sprint 1

Sprint one focused on data pre-processing components of the system. The goal of sprint one was to complete backlog items 'load dataset', 'visualise data' and 'apply signal filter'. The backlog item 'create storage class for 'datasets' was created after the sprint was started. At the end of the sprint all goals were met, including the added backlog item. Some time was also spent working on the backlog item 'refactor code' but is not counted in the planned work for the sprint.

Sprint overview

Planned		Completed	
Items	Time	Items	Time
3	9h	4	9h

Burndown chart for sprint 1



Backlog at the end of sprint 1

ID	Title	Status	Estimate Time	Actual Time
1	Load dataset	Complete	3	1
2	Visualise data	Complete	3	3
3	Filter noise	Complete	3	2
4	Extract features	TODO	4	0
5	Prepare training	TODO	3	0
6	Train KNN	TODO	1	0
7	Train SVM	TODO	1	0
8	Test ML models	TODO	4	0
9	Refactor code	In Progress	5	1
10	Store dataset	Complete	2	2

Sprint 2

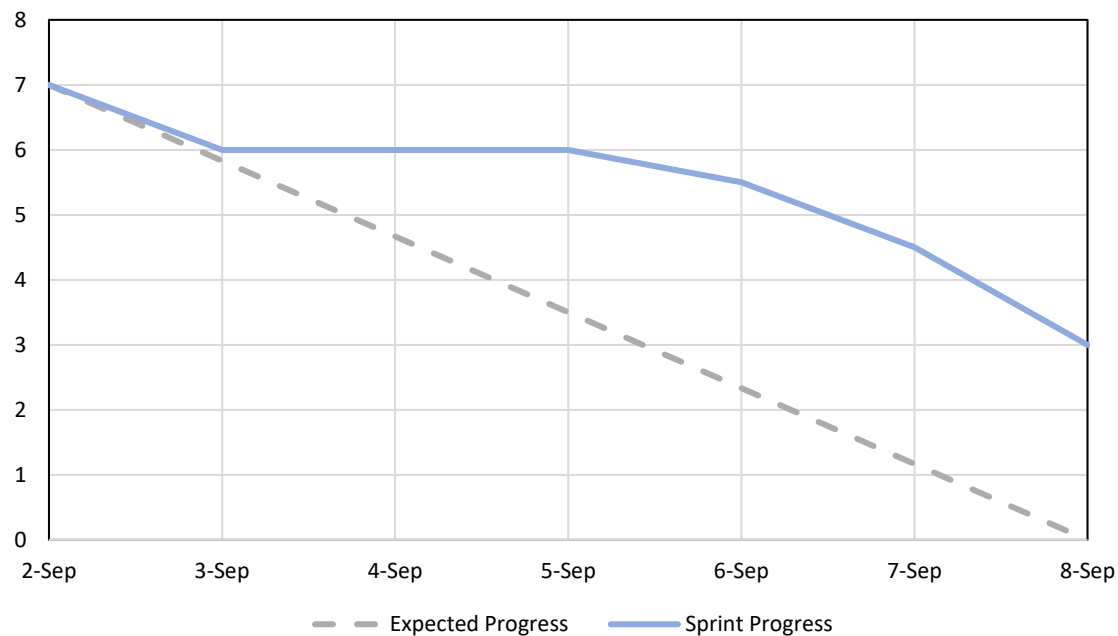
Sprint 2 focused on the feature preparation components of the system. The goal of the sprint was to complete backlog items 'Extract features' and 'Prepare training sets'.

Unfortunately, at the end of this sprint both items were still incomplete and time must be made for them to be completed in the next sprint. Time was also spent on backlog item 'refactor code' but is not counted in the planned work for the sprint.

Sprint overview

Planned		Completed	
Items	Time	Items	Time
2	7h	0	4h

Burndown chart for sprint 2



Backlog at the end of sprint 2

ID	Title	Status	Estimate Time	Actual Time
1	Load dataset	Complete	3	1
2	Visualise data	Complete	3	3
3	Filter noise	Complete	3	2
4	Extract features	In Progress	4	2
5	Prepare training	In Progress	3	2
6	Train KNN	TODO	1	0
7	Train SVM	TODO	1	0
8	Test ML models	TODO	4	0
9	Refactor code	In Progress	5	3
10	Store dataset	Complete	2	2

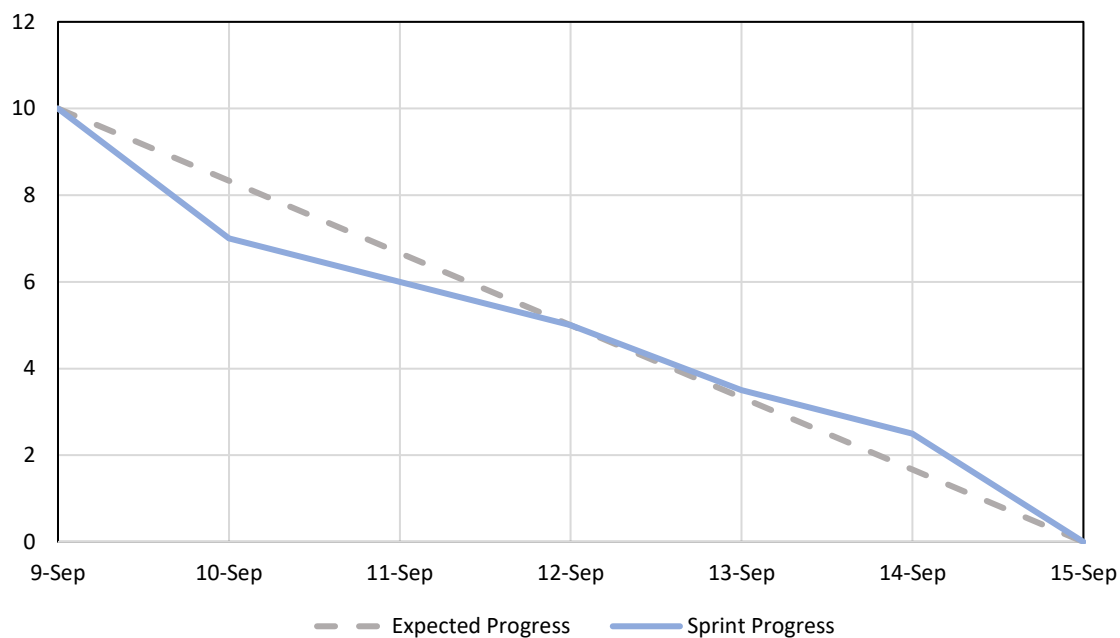
Sprint 3

Sprint 3 focused on training and evaluating the machine learning classifiers, as well as completing the system as a whole. The goals for this sprint include completing backlog items 'Train KNN', 'Train SVM', 'Test ML models' and 'Refactor code'. Time also had to be set aside to complete items left from sprint 2. Despite this all backlog items were completed by the end of the sprint

Sprint overview

Planned		Completed	
Items	Time	Items	Time
6	10h	6	9h

Burndown chart for sprint 3



Backlog at the end of sprint 3

ID	Title	Status	Estimate Time	Actual Time
1	Load dataset	Complete	3	1
2	Visualise data	Complete	3	3
3	Filter noise	Complete	3	2
4	Extract features	Complete	4	3
5	Prepare training	Complete	3	3
6	Train KNN	Complete	1	1
7	Train SVM	Complete	1	1
8	Test ML models	Complete	4	3
9	Refactor code	Complete	5	5
10	Store dataset	Complete	2	2

Implementation

Butterworth filter

The component 'Filter noise' implements a fourth order lowpass Butterworth filter which permits frequencies below 0.04. Butterworth filters are commonly used for the analysis of motion because they provide a maximally flat response in the pass band; it flattens out ripples in the data making it easier to analyse transitions in activities

Time Series

The data from each activity represents a time series, where each entry is a change in movement recorded at a particular time. In order to extract meaningful features from the data set it must first be processed into segments. The technique used in 'Extract features' divides an activity into time windows, which is about 1000 entries. Features are then extracted from the time window, providing the feature set with the characteristics of an activity at different times in its performance.

KNN Classifier

Knn stands for K-nearest neighbours, where a data point is classified based on the classification of a select amount of its nearest neighbouring data points. KNN is one of the machine learning models evaluated in making predictions on human activities. The KNN model used has K set to 3, as a higher or lower setting resulted in a lower accuracy score.

SVM Classifier

The modelling technique used, SVM stands for support vector machine. The central function of an SVM classifier is finding a 'maximum marginal hyperplane' that optimally divides the dataset into distinct classes. Within an SVM context, Data points are known as support vectors. Calculations are made, based off a support vector's closeness to the hyperplane, to better define the distinct features of a class.

Evaluation

[See appendix for accuracy scores and confusion plots for each test]

To evaluate the accuracy of the two machine learning models four tests were run, where a model was trained using datasets with a set number of features extracted from a specific number of sensors. Analysis of a model's accuracy was performed using an accuracy score and plotting a confusion matrix for each test.

Test one trained both models on a feature set where the minimum, maximum and average values were extracted from three data columns; the x, y and z axes for the wrist accelerometer. This totalled at 9 features per entry. Despite using the features of a small part of the dataset, both classifiers were relatively accurate in predicting activities. The KNN classifier scored 0.75, while SVM scored slightly lower at 0.71.

Test two used the same data columns but extracted three more features in addition to the minimum, maximum and average. The added features were the Standard deviation, Skewness and Kurtosis of each column, with a total of 18 features per entry. The added features made little difference to KNN score, only increasing to 0.7562. But it made a noticeable difference to the SVM score which increased to 0.78, making it more accurate than the KNN model.

Test three extracted the minimum, maximum and average features from all data columns in the data set. This totalled to 72 features per entry. This drastic increase to the volume of data was reflected in the improvement to the accuracy scores of both models. KNN scored 0.92, while SVM scored 0.93 remaining the more accurate model by a small margin.

Test four extracted all features (Min, max, mean, standard deviation, skewness, kurtosis) from all data columns. This totalled to 144 features per entry. While doubling the amount of data from test three increased the SVM accuracy score to 0.945, the KNN accuracy score decreased to 0.912.

Based on the accuracy metrics of each test it appears that SVM is a better candidate algorithm for human activity recognition projects. KNN appears to perform more accurately with less data. But that performance can reverse when it is trained with more detailed feature sets. While SVM performed worse than KNN with smaller data sets, it made more accurate predictions when it was provided with larger and more detailed feature sets. However, improvements can still be made. All models in each test made some inaccurate predictions, especially in attempting to classify Bicycling on ergometer 50W and Bicycling on ergometer 100W (tasks 11 and 12).

References

[1] Jason Brownlee, "A Gentle Introduction to a Standard Human Activity Recognition Problem", 2018 [Online]

Available: <https://machinelearningmastery.com/how-to-load-and-explore-a-standard-human-activity-recognition-problem/>

[Accessed 08 September 2019]

[2] Pierluigi Casale, Oriol Pujol, Petia Radeva, "Human Activity Recognition from Accelerometer Data Using a Wearable Device", Sept, 2011 [Online]

Available: https://link.springer.com/chapter/10.1007/978-3-642-21257-4_36

[Accessed 08 September 2019]

[3] Bill McNeese, "Are the Skewness and Kurtosis Useful Statistics?", 2016 [Online]

Available: <https://www.spcforexcel.com/knowledge/basic-statistics/are-skewness-and-kurtosis-useful-statistics>

[Accessed 08 September 2019]

























[4] Quickscrum, "Sprint Burn Down Chart", 2019 [Online]

Available: <https://www.quickscrum.com/ScrumGuide/183/sg-Sprint-Burn-Down-Chart>

[Accessed 09 September 2019]

Appendix

Version control

Branch: master ▾		
Commits on Sep 12, 2019		
Scores from ML accuracy tests, changed some variable names 42089069 committed 4 days ago	 a82c38a	
Commits on Sep 11, 2019		
feature_engineering_example() separated into new functions 42089069 committed 4 days ago	 1b4a269	
Separated knn & svm into 2 functions 42089069 committed 5 days ago	 abc7e72	
Add function to load training data sets 42089069 committed 5 days ago	 8142e71	
Commits on Sep 9, 2019		
Copy model_training_and_evaluation_example() from har.py 42089069 committed 7 days ago	 fe086b5	
Commits on Sep 8, 2019		
Import stats for skew and kurtosis 42089069 committed 7 days ago	 ac09763	
Add informal tests 42089069 committed 7 days ago	 5b92594	
Add new feature_engineering_example() 42089069 committed 7 days ago	 a9769a5	
Copied feature_engineering_example() from har.py, training and test c... 42089069 committed 7 days ago	 167f409	
Add function for filtering data, informal tests for function 42089069 committed 7 days ago	 39ae807	
Added class dataSets() for data handling 42089069 committed 7 days ago	 f711234	
Commits on Aug 26, 2019		
Initialise Repository 42089069 committed 20 days ago	 5dfd89a	

Test scores

	KNN	SVM
3 sensors, 3 features	Accuracy: 0.7504798464491362	Accuracy: 0.710172744721689
	[43 3 0 0 0 1 0 0 0 0 7 3 0]	[35 3 0 0 0 0 0 0 0 0 16 3 0]
	[3 54 0 0 0 0 0 0 0 0 0 0 0]	[6 51 0 0 0 0 0 0 0 0 0 0 0]
	[0 3 48 1 2 0 1 0 0 0 1 1 0]	[0 3 48 1 1 0 2 0 0 0 1 1 0]
	[2 0 2 84 4 5 1 1 0 0 0 0 0]	[2 0 2 84 6 3 1 0 0 0 0 1 0]
	[0 0 1 8 36 7 1 3 0 0 1 0 0]	[0 0 5 5 36 6 4 0 0 0 0 1 0]
	[1 1 2 14 26 32 4 1 1 0 1 2 0]	[0 1 3 7 27 35 9 0 1 0 0 2 0]
	[1 0 6 4 2 6 191 4 5 0 0 0 0]	[0 0 3 5 1 8 202 0 0 0 0 0 0]
	[0 0 1 0 0 2 14 21 1 0 0 0 0]	[0 0 1 0 0 1 37 0 0 0 0 0 0]
	[0 0 0 1 0 1 9 2 24 0 1 0 0]	[0 0 2 2 0 2 18 0 13 0 1 0 0]
	[0 0 0 0 0 0 0 0 0 0 96 0 0]	[0 0 0 0 0 0 0 0 0 0 96 0 0]
	[0 0 1 3 2 7 2 0 0 0 58 27 0]	[3 0 0 2 0 9 2 0 0 0 70 14 0]
	[0 0 1 1 0 0 0 0 0 0 39 59 0]	[0 0 0 2 0 0 0 0 0 0 64 34 0]
	[0 0 0 1 0 0 1 0 0 0 0 0 36]	[0 0 1 0 0 1 0 0 0 0 0 0 36]
3 sensors, 6 features	Accuracy: 0.7562380038387716	Accuracy: 0.7802303262955854
	[40 7 1 1 0 0 0 0 0 0 4 4 0]	[46 4 1 1 0 0 0 0 0 0 3 2 0]
	[5 51 0 1 0 0 0 0 0 0 0 0 0]	[6 51 0 0 0 0 0 0 0 0 0 0 0]
	[0 4 47 1 0 0 2 1 0 0 1 1 0]	[1 3 48 1 1 0 1 0 0 0 1 1 0]
	[1 1 2 78 7 7 2 0 0 0 0 1 0]	[2 0 2 83 3 8 0 0 0 0 1 0 0]
	[0 0 1 10 32 7 2 4 0 0 0 1 0]	[0 0 3 7 36 9 1 1 0 0 0 0 0]
	[2 0 1 12 25 34 7 2 1 0 0 1 0]	[0 1 2 6 23 46 5 0 1 0 0 1 0]
	[0 0 2 1 3 7 197 6 3 0 0 0 0]	[0 0 2 1 1 7 199 6 3 0 0 0 0]
	[0 0 1 0 0 0 13 23 2 0 0 0 0]	[0 0 1 0 0 0 11 25 2 0 0 0 0]
	[0 0 0 1 1 1 7 1 26 0 1 0 0]	[0 0 1 0 1 3 7 2 24 0 0 0 0]
	[0 0 0 0 0 0 0 0 0 0 96 0 0]	[0 0 0 0 0 0 0 0 0 0 96 0 0]
	[2 0 0 3 1 6 3 0 0 0 68 17 0]	[3 0 0 2 1 8 2 0 0 0 73 11 0]
	[2 0 2 1 0 0 0 0 0 0 36 59 0]	[1 0 0 1 0 0 0 0 0 0 47 51 0]
	[0 0 0 0 0 0 1 0 0 0 0 0 37]	[1 0 1 0 0 0 1 0 0 0 0 0 35]
24 sensors, 3 features	Accuracy: 0.9222648752399232	Accuracy: 0.9309021113243762
	[57 0 0 0 0 0 0 0 0 0 0 0 0]	[57 0 0 0 0 0 0 0 0 0 0 0 0]
	[0 57 0 0 0 0 0 0 0 0 0 0 0]	[0 57 0 0 0 0 0 0 0 0 0 0 0]
	[0 0 56 1 0 0 0 0 0 0 0 0 0]	[0 0 57 0 0 0 0 0 0 0 0 0 0]
	[0 0 2 96 1 0 0 0 0 0 0 0 0]	[0 0 1 96 1 1 0 0 0 0 0 0 0]
	[0 0 1 1 47 5 1 1 1 0 0 0 0]	[0 0 1 1 47 8 0 0 0 0 0 0 0]
	[0 0 0 1 16 67 0 1 0 0 0 0 0]	[0 0 0 1 7 77 0 0 0 0 0 0 0]
	[0 0 2 0 3 1 211 1 0 0 1 0 0]	[0 0 2 0 3 4 210 0 0 0 0 0 0]
	[0 0 1 0 0 0 2 35 0 0 0 1 0]	[0 0 0 0 1 1 0 37 0 0 0 0 0]
	[0 0 0 0 0 1 2 0 35 0 0 0 0]	[0 0 0 0 0 2 0 0 36 0 0 0 0]
	[0 0 0 0 0 0 0 0 1 95 0 0 0]	[0 0 0 0 0 1 0 0 0 95 0 0 0]
	[0 0 0 0 0 1 0 0 0 0 82 17 0]	[0 0 0 0 0 3 0 0 0 0 84 13 0]
	[0 0 0 0 0 0 1 0 1 0 12 86 0]	[0 0 0 0 0 2 0 0 0 0 17 81 0]
	[0 0 0 0 0 1 0 0 0 0 0 0 37]	[0 0 0 0 0 2 0 0 0 0 0 0 36]
24 sensors, 6 features	Accuracy: 0.9126679462571977	Accuracy: 0.9452975047984645
	[52 1 2 2 0 0 0 0 0 0 0 0 0]	[56 0 0 1 0 0 0 0 0 0 0 0 0]
	[0 57 0 0 0 0 0 0 0 0 0 0 0]	[0 57 0 0 0 0 0 0 0 0 0 0 0]
	[3 0 52 2 0 0 0 0 0 0 0 0 0]	[0 0 57 0 0 0 0 0 0 0 0 0 0]
	[0 0 3 96 0 0 0 0 0 0 0 0 0]	[0 0 2 95 1 1 0 0 0 0 0 0 0]
	[0 0 1 2 49 5 0 0 0 0 0 0 0]	[0 0 1 2 48 6 0 0 0 0 0 0 0]
	[0 0 0 2 22 58 1 2 0 0 0 0 0]	[0 0 0 1 8 76 0 0 0 0 0 0 0]
	[0 0 0 1 5 1 212 0 0 0 0 0 0]	[0 0 2 0 4 2 210 0 1 0 0 0 0]
	[0 0 1 0 0 0 1 36 0 0 0 1 0]	[0 0 1 0 0 0 0 38 0 0 0 0 0]
	[0 0 0 0 0 1 1 0 36 0 0 0 0]	[1 0 0 0 0 1 0 0 36 0 0 0 0]
	[0 0 0 0 0 0 0 0 0 96 0 0 0]	[0 0 0 0 0 1 0 0 95 0 0 0]
	[0 0 0 0 0 1 0 0 0 0 86 13 0]	[0 0 1 0 0 2 0 0 0 0 88 9 0]
	[0 0 0 0 0 0 1 0 1 0 14 84 0]	[0 0 1 0 0 1 0 0 0 0 6 92 0]
	[0 0 0 0 0 0 1 0 0 0 0 0 37]	[0 1 0 0 0 0 0 0 0 0 0 0 37]