ML Project Notes – OBD Data Analysis

I resume this project after some months of break.

First thing I’ll do is to create an artificial dataset in order to have clearly separable data. For now, I give up the idea of recognizing who is driving. I’m going to try to detect if a driving session is related to an itinerary of type:

* Urban
* (Out-of-town)
* Highway
* Combined

Let’s start from a two-class classification:

City:

* Lower speeds
* More variable speed (traffic)
* More gear changes
* More acceleration (from a standing still up to speed limit)
* More braking (often to a dead stop at intersections, traffic lights and stop signs)

Highway:

* Higher speeds
* More constant speeds
* Less gear changes
* Less acceleration
* Less braking

So, I could measure driving session of about 5 minutes. I’ll have two timeseries: RPM and speed. I can extract the following features:

* Mean speed <v>, maximum speed V, minimum speed v;
* Mean acceleration <a>, max acceleration A, min acceleration a (in module)
* Number of gear changes #GC

It could be good to have also data about throttle position and brakes, but firstly I should find out if my car supports them, secondly, they would increase the sampling time.

NOTE – I put all the files related to the driver recognition in a dedicated folder Old\_DriverRecognition. These files are:

* *main\_OldDriverRecognition*
* *OrganizeCollectedData\_OldDriverRecognition*

In the *main* script I try now to extract the features. I still use the *ComputeFirstDerivative* written some months ago, but I think I should revise it. In particular, in *ComputeGearShifts* I don’t obtain beautiful results: it often happens that a decrease in the number of RPMs is so long that it counts for more than one gear shifts. For now, I continue with this behaviour in order to arrive at the design of the classifier.

So, I can create my dataset. I have 7 features, so it’s seven-dimensional.

The first trial I do is with the *kernel regularized least squares* classifier. My dataset is very small and so the test is not so significant, but it can give me an idea about what’s happening. With 9 driving sessions, three of which are dummy, the error on the little test set is zero. I absolutely have to collect more and more data. In the meantime, I try the other algorithms of classification. I’ll also have to do some cross-validation, remember it!

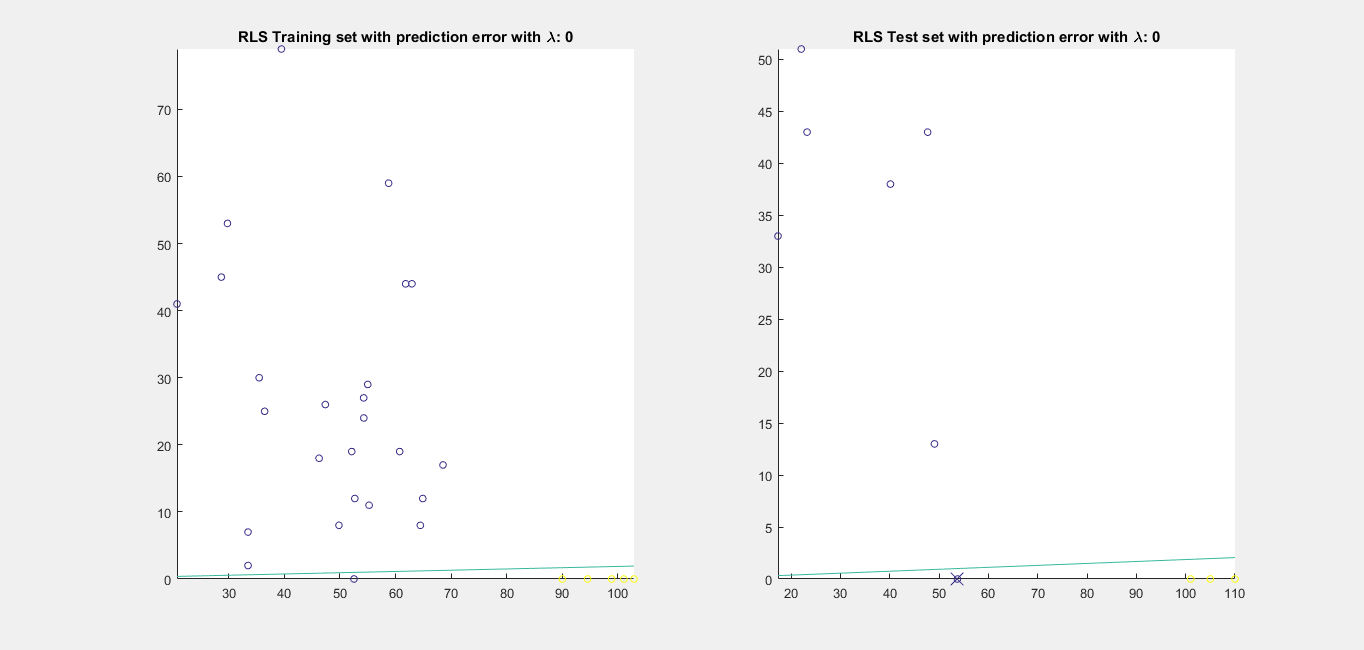
*13/05*

I add some examples to my dataset, some are taken on highway drive, other from out-of-town drive. One of the most interesting example is the one recorder on “sopraelevata”: here there aren’t gear shifts, but the speed is almost constant; I label it as out-of-town and I’m curious to know how the classifier will manage it.

14/06

My dataset is getting a decent size. I begin to collect some screenshot about interesting issues.

Image “RLS\_BinaryClassification\_OneError\_NoOffset”:



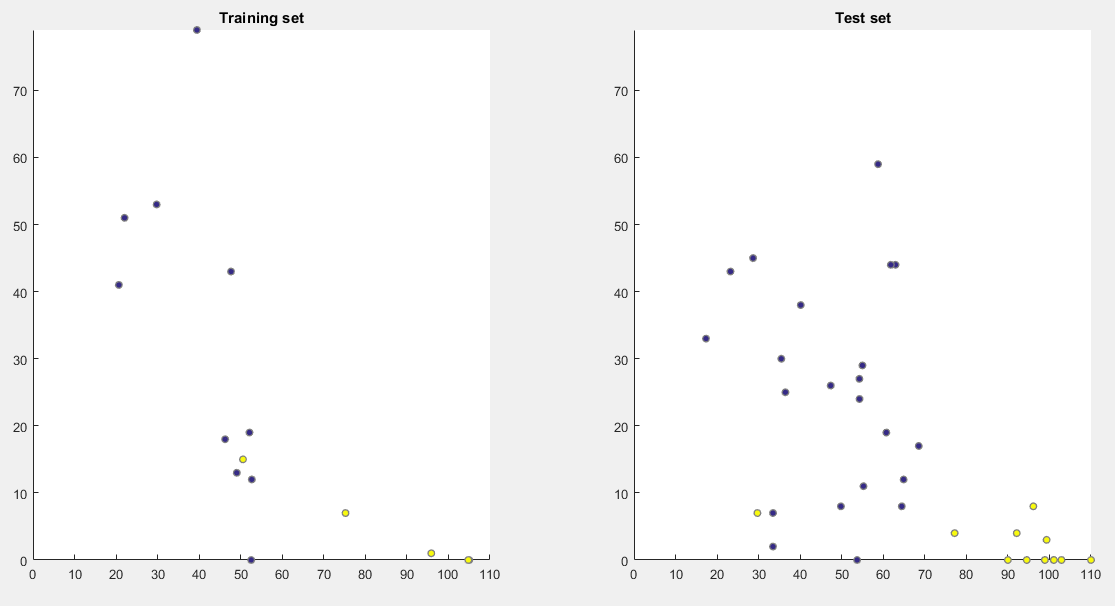
Here I can see that the classifier cannot avoid to do an error already in the training set: the hypothesis is linear and it must pass through the origin. There is an example which belongs to “out-of-town”, but it’s probably taken on *sopraelevata*, so the number of gear changes is 0 and there doesn’t exist a linear function that can separate it from other point with zero gear changes. This suggests me to use a linear function with offset.

So, at a first instance RLS didn’t give me beautiful results: without offset it was impossible to find a function capable of distinguish session on the basis of just mean speed. With the management of the offset, things go much better. In fact, the data are clearly linearly separable, linearly in the sense of linear function plus constant.

An interesting discover has been that the considerations about dealing with offset aren’t valid with logistic regression: probably the solution of the minimization problem over w and b is not so simple when considering other loss function than square loss.

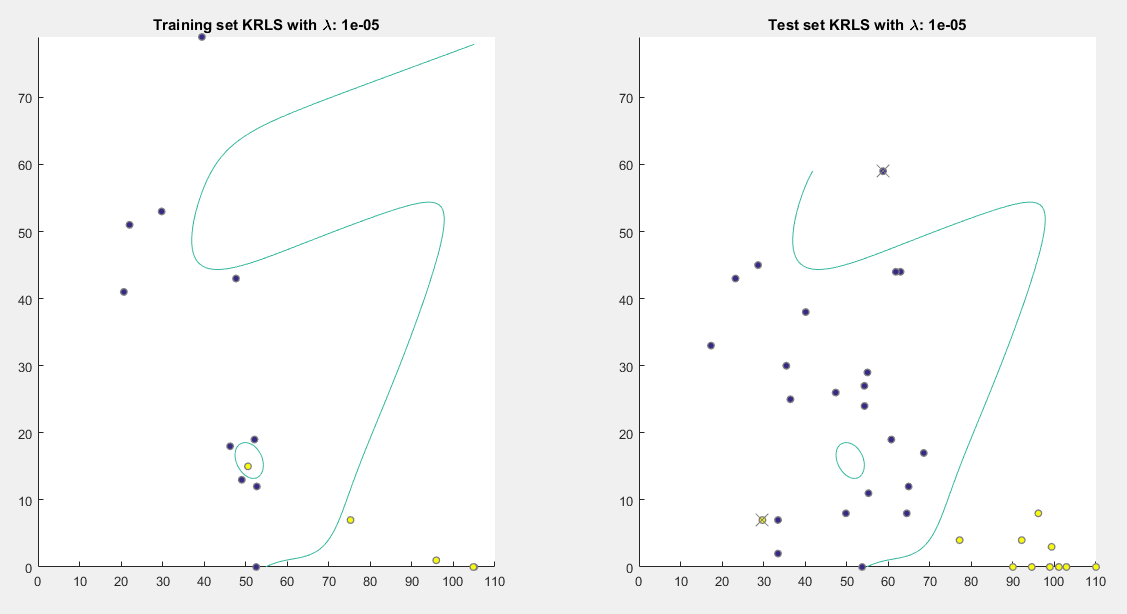
15/06

Adding some data from Ennio driving session I obtained a very interesting dataset:

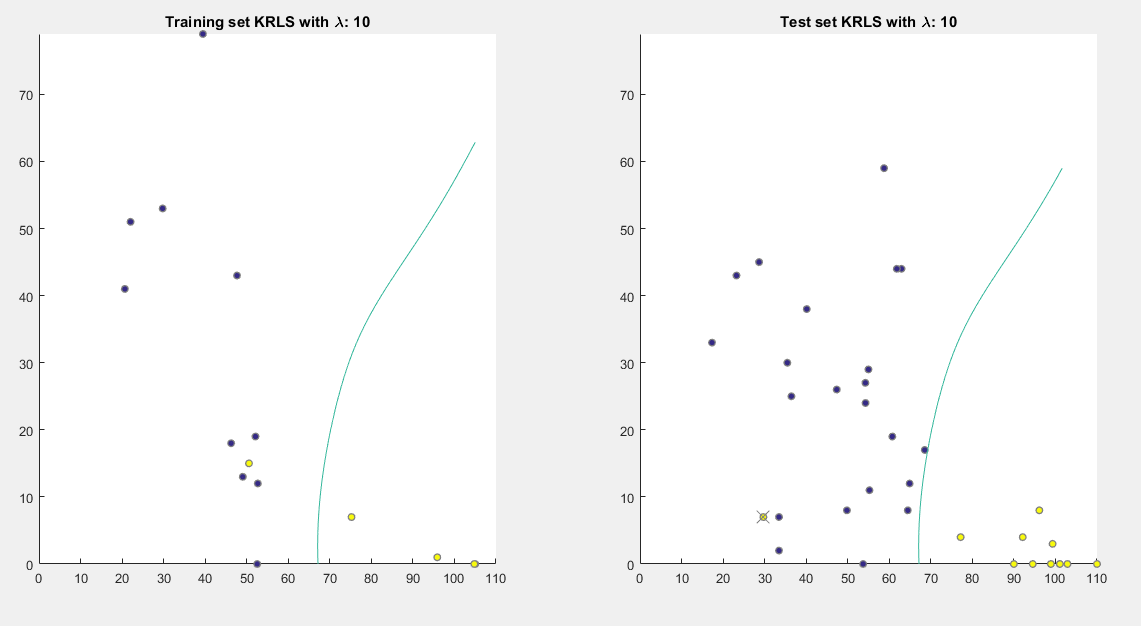


I can see that there is an outlier in both training and data set. I’m curious to see what will happen.

With KRLS and lambda=1e-5:

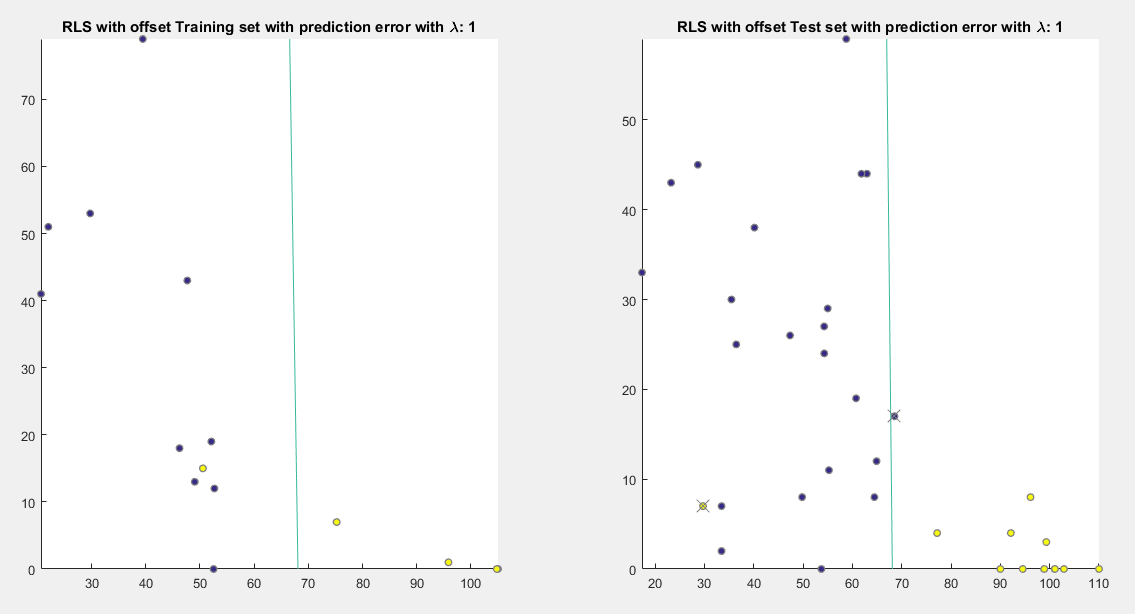


The predictive function is quite absurd. Increasing lambda, the solution should be more stable. In effect:



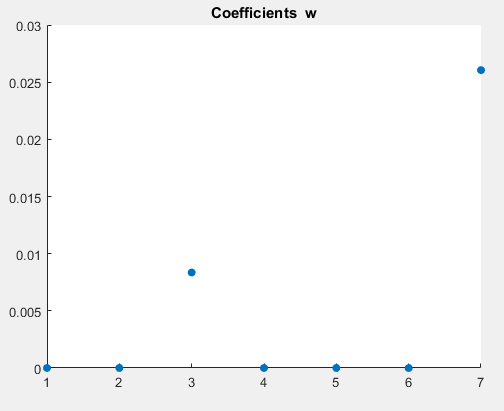
Obviously, the outlier in the test set is not predicted well, but it seems reasonable.

So, using KRLS for these data that are clearly linearly separable could be seen as “use a sledgehammer to crack a nut”, but it made me to see how important the regularization in an actual case is.

Now, I perform the same analysis with RLS:

Here the problem doesn’t subsist for any value of lambda: the outlier is however ignored thanks to square loss and hypothesis space.

Then, I made some tests using all the seven variables. In this case the RLS without offset achieves zero-errors for any lambda. The RLS with offset achieves zero errors with sufficient big lambda. I also run Orthogonal Matching Pursuit, with cross validation to find the best T, and the result was interesting:



The result is good: two of the most important variables are minSpeed and numberOfGearShifts. Maybe the choice is influenced by the random division of training set, because I suppose that there are some measures in which the mean speed would be better than the minimum one, but the result in this case could be expected, anyway. In effect, running several times the cross-validation gives different results: maybe the dataset is too little.

17/06

With regard to the last paragraph, I think that the key result I get from OMP is that, depending on the training set, there are always just few coefficients different from zero: this is because the most of the times they are sufficient just some of the seven dataset variables to identify driving condition.

I wrote a new file, *MultiClassClassificationModel.m*, in which I perform a three classes classification. The three classes are:

* 1 – out-of-town
* 2 – highway
* 3 – city

The dataset variables are the same of the binary classifier. Firstly, I tried a *one-vs-all* RLS: a linear model cannot describe the out-of-town class points, which are in the middle of the other two classes, so the final error is high. Indeed, I proceeded with *one-vs-all* KRLS and I got better results. I think they could be even better if dataset was richer.

Now I write the README file for the git repository.

This repository contains all the files I used to perform the analysis of the data I collected from several driving sessions. Data are about the speed and the RPM (revolutions per minutes). To collect them I developed the app OBDConnection whose code can be found on my GitHub page (it basically establishes a Bluetooth connection between a smartphone and a car, through an OBD II Bluetooth adapter).

The analysis goal is to infer some information about the driving conditions of the various driving sessions. In particular, it should tell in what type of road they were taken, e.g. highway, out-of-town, city (in-town). At the beginning, I also wanted to infer who was the driver of each session, but after having written some pre-processing code (contained in the folder OBDDataAnalysis\Old\_DriverRecognition) I decided to focus on a simpler problem.

The MATLAB project is contained in OBDDataAnalysis\project, while all the other folders contains the measured data, divided into sessions. Inside the project folder you can find all the .m files. In particular:

* “OrganizeCollectedData.m” contains the code to read the collected data and save them in a first raw dataset, in which each row is a timeseries (of speed and of RPM);
* “main.m” contains the code to load this raw dataset and build the actual dataset, with seven variables obtained from the various timeseries;
* “ClassificationModel.m” contains the code to classify sessions in highway sessions or out-of-town sessions; I used several classification algorithms (RLS, RLS with offset, KRL, LR, OMP), each of one can be run within a dedicated section;
* “MultiClassClassificationModel.m” contains the code to classify sessions in highway, out-of-town or city sessions; here there two section, one to run one-vs-all RLS and one for one-vs-all KRLS.

All the other files are needed for the various classifiers to work (these are quite all taken from the laboratory sessions of Machine Learning course held at the University of Genoa) or for the feature extraction from the measured data, e.g. to compute acceleration from speed data.