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!

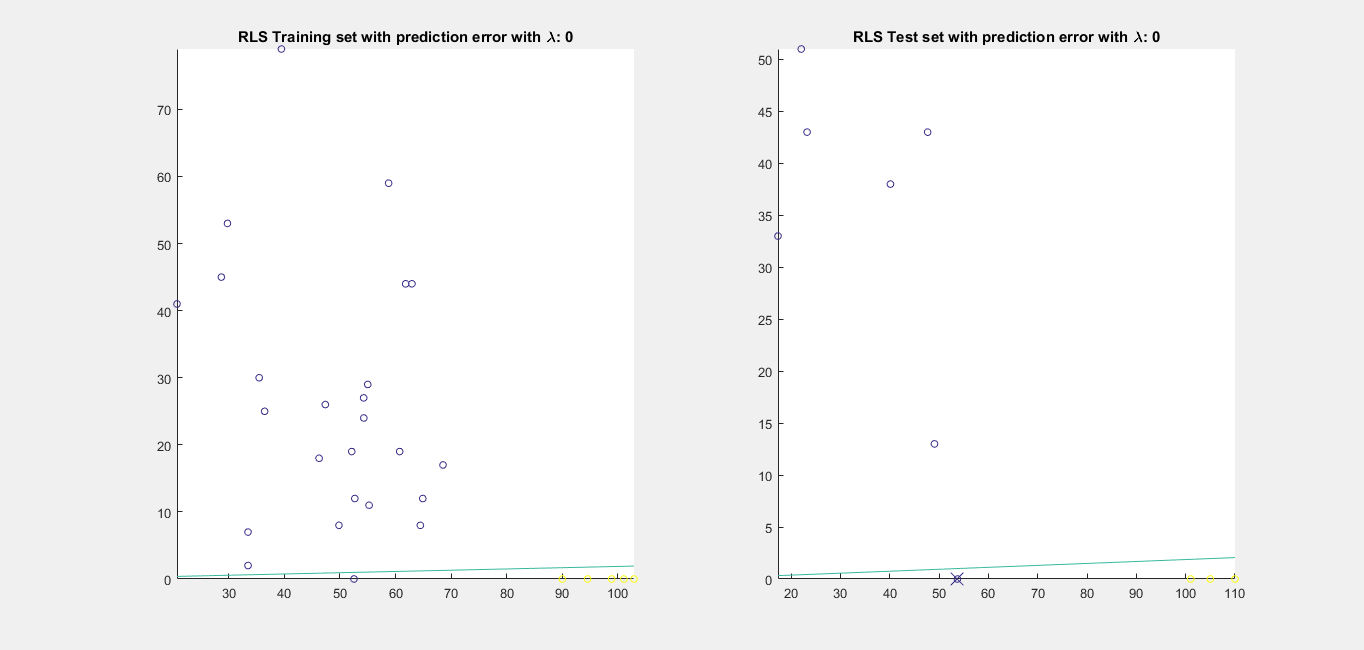
*13/05*

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”:



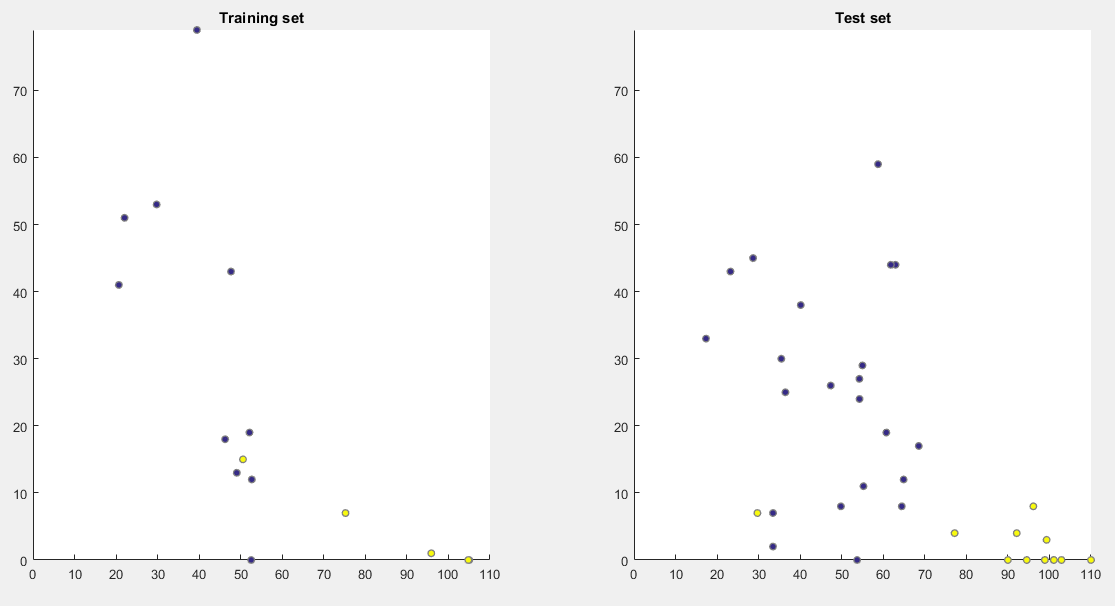
Here I can see that the classifier cannot avoid to do an error already in the training set: the hypothesis is linear and it must pass through the origin. There is an example which belongs to “out-of-town”, but it’s probably taken on *sopraelevata*, so the number of gear changes is 0 and there doesn’t 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.

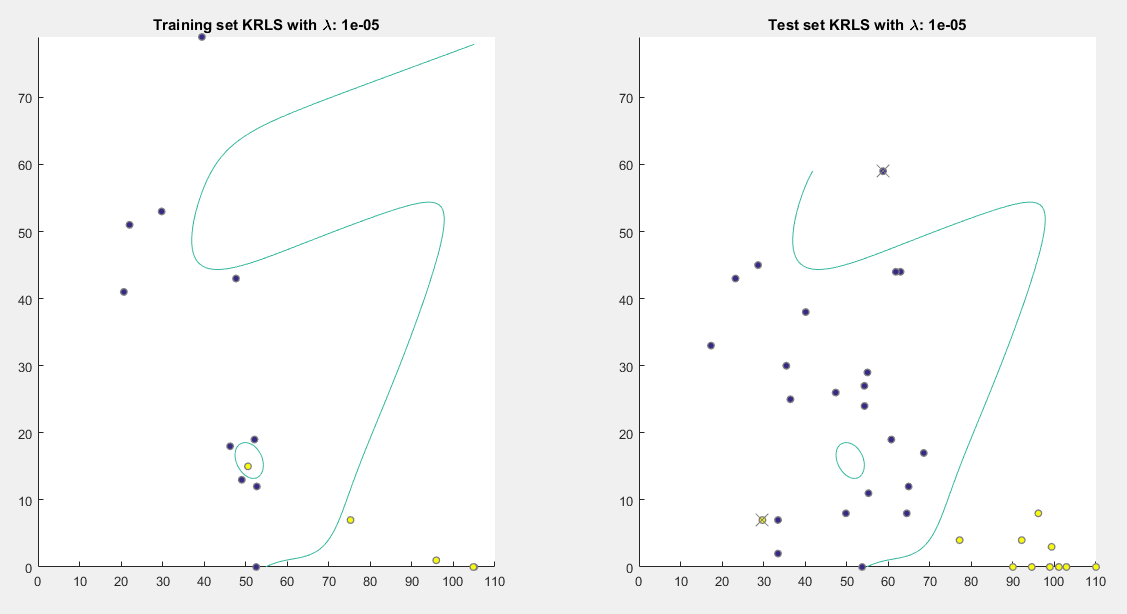
15/06

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

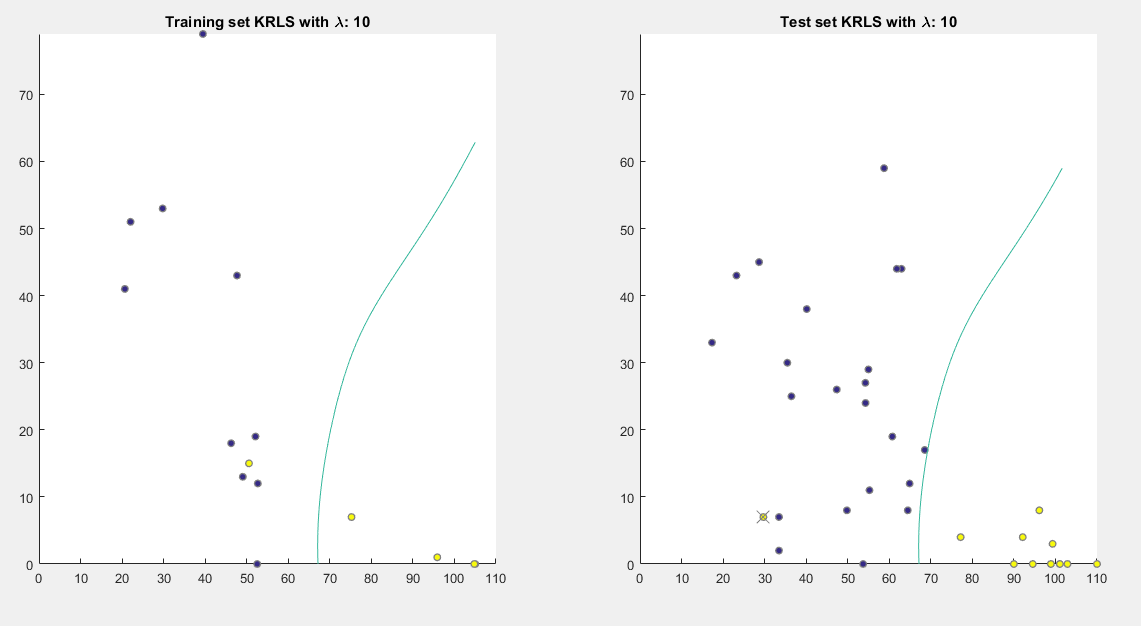


I can see that there is an outlier in both training and data set. I’m curious to see what will happen.

With KRLS and lambda=1e-5:

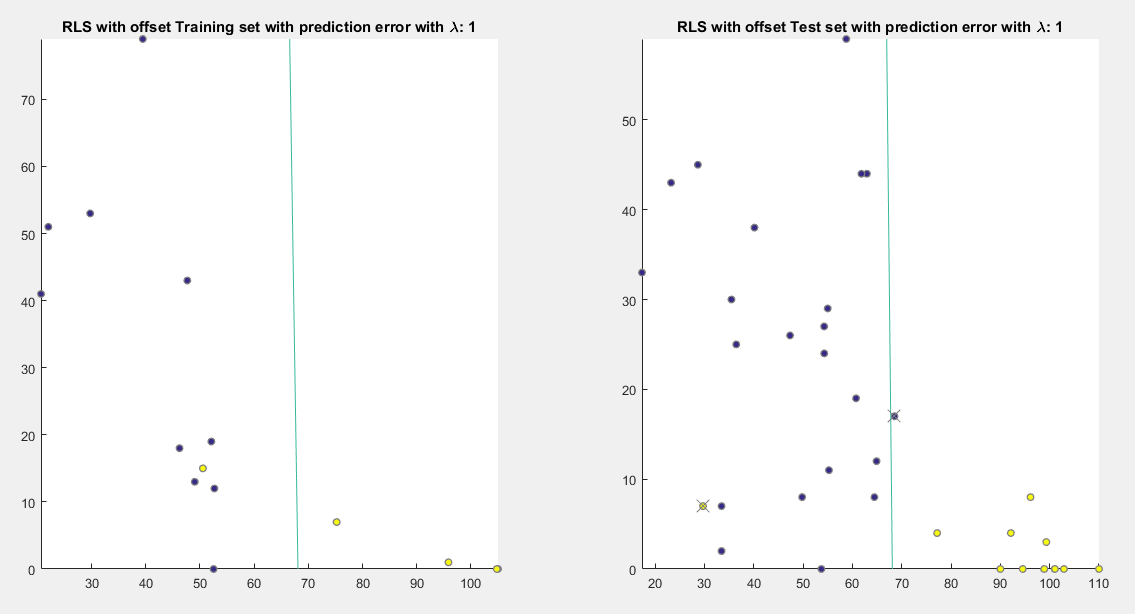


The predictive function is quite absurd. Increasing lambda, the solution should be more stable. In effect:



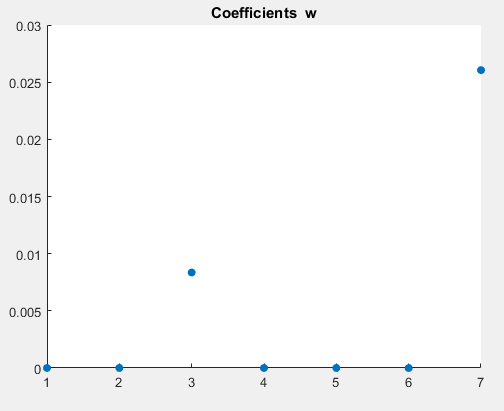
Obviously, the outlier in the test set is not predicted well, but it seems reasonable.

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.