Data analysis and data Mining
Report 1

Mauro Angelini
Alessio Gilardi

November 4, 2018

Contents

First exercise **2**

Second exercise **4**

Third exercise **6**

Real world problem **8**

First exercise

In this first exercise we implemented the regression of a function in one-dimensional case, for example: the quadratic function

$$y = x^2, \ \forall x \in [0, 1]$$

Vengono definite:

- Function to be rebuilt
- The number of samples
- The samples disturbed by the Gaussian additive noise
- The variance of the noise
- The degree of the polynomial regressor

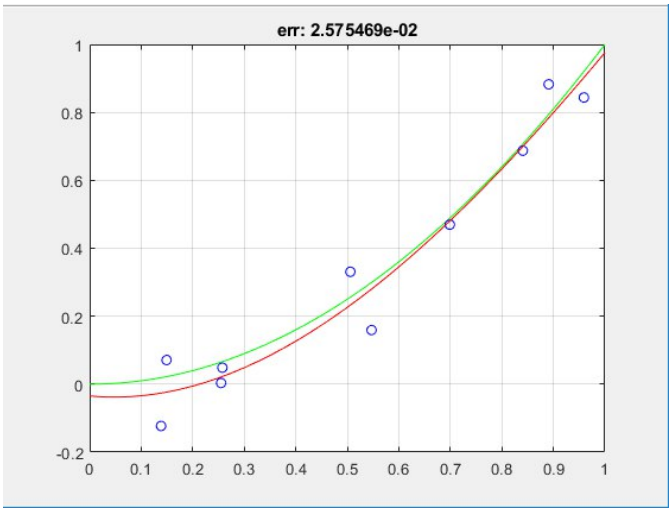
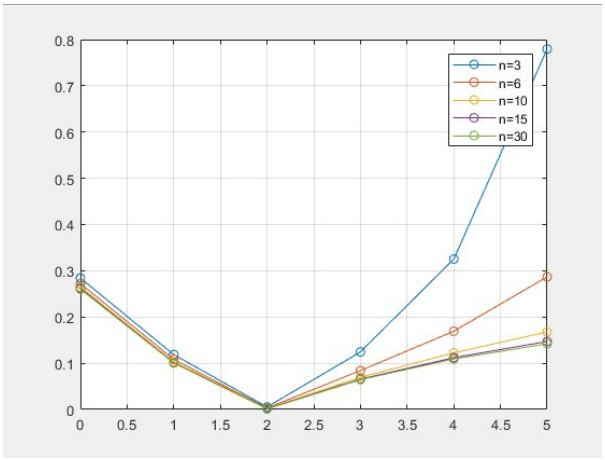


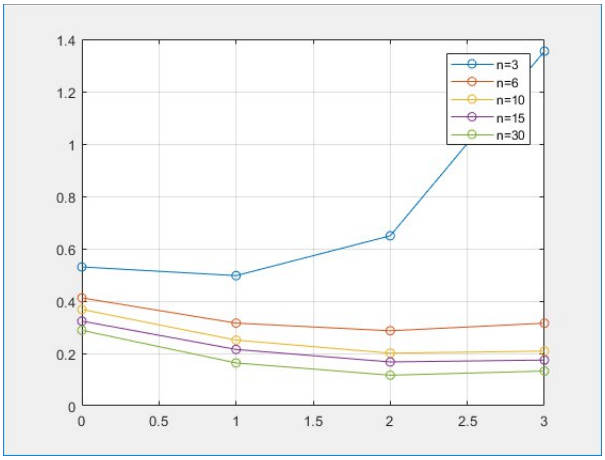
Figure 1: Regression Function

Figure 1 Regression function of a quadratic function
Starting from this first test, we notice that increasing the complexity of the polynomial too much produce a great loss of precision in the reconstruction. It's therefore necessary to evaluate the variation of reconstruction, by changing n (Number of samples), p (Degree of the polynomial regressor) and σ (Noise variance). So the error is calculated between YT (True function) and YP (Prediction function) and plotted in the graph.
Now we notice that in order to make the measurement of error more precise, it's better to repeat the experiment a sufficiently high number of times, for example, we make 30 repetitions of the prediction.

We performed tests with some values of n and p, from which it becomes clear how the error always maintains the same trend. In particular, observing the error plots, it's evident that the maximum degree of precision is obtained



(a) Error by varying n and p (from 0 to 5)



(b) Error by varying n and p (from 0 to 3)

Figure 2: Comparing error with different values of p

	p = 0	p = 1	p = 2	p = 3	p = 4	p = 5
n = 3	2.842e-01	1.186e-01	5.349e-03	1.245e-01	3.253e-01	7.796e-01
n = 6	2.719e-01	1.084e-01	3.093e-03	8.410e-02	1.693e-01	2.869e-01
n = 10	2.647e-01	1.030e-01	2.056e-03	6.957e-02	1.225e-01	1.677e-01
n = 15	2.622e-01	1.012e-01	1.514e-03	6.572e-02	1.120e-01	1.471e-01
n = 30	2.603e-01	1.004e-01	1.095e-03	6.449e-02	1.092e-01	1.421e-01

(a) Error by varying n and p (from 0 to 5)

	p = 0	p = 1	p = 2	p = 3
n = 3	5.303e-01	4.972e-01	6.493e-01	1.354e+00
n = 6	4.121e-01	3.157e-01	2.865e-01	3.154e-01
n = 10	3.678e-01	2.507e-01	2.010e-01	2.089e-01
n = 15	3.235e-01	2.152e-01	1.674e-01	1.746e-01
n = 30	2.882e-01	1.636e-01	1.168e-01	1.325e-01

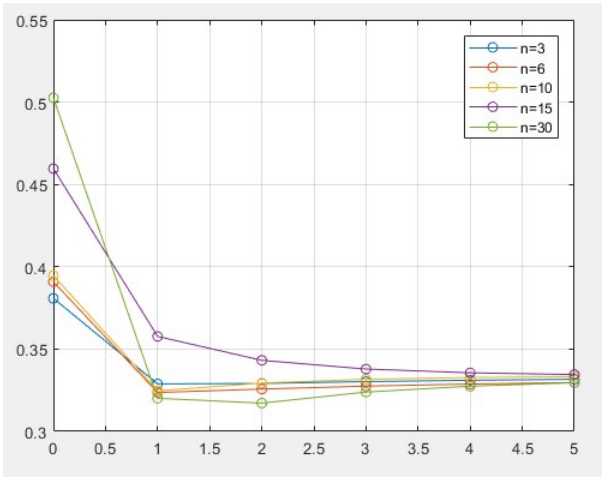
(b) Error by varying n and p (from 0 to 3)

Figure 3: Comparing error with different values of p

for p values around the degree of the original function. Obviously, using a second degree polynomial the error is minimal since the function to be reconstructed is a parabola. Utilizing a large number of samples n, in general, also increases accuracy, while an increase of p results in a loss of precision. It's evident that the variation of the error between one execution of the exercise and the other is minimal for the values greater than n. While using a small number of samples n, it's better to use low-grade polynomials, since an excessive number of degrees of freedom results in an attempt of the regression function to pass for each sample, making the reconstruction less precise, in particular in presence of a lot of noise, as it will try to pass for the samples even if noisy.

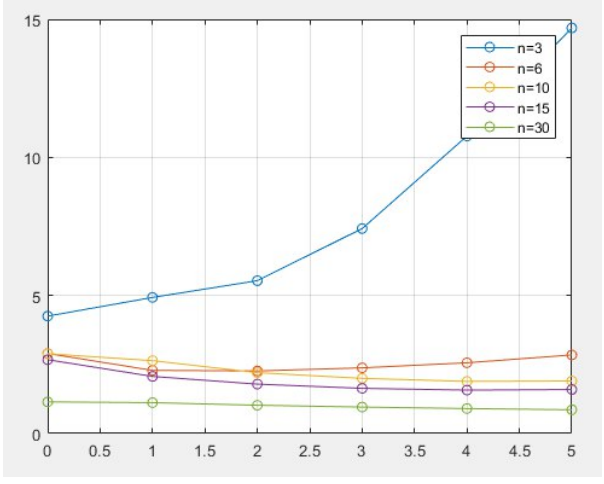
Second exercise

We repeat the polynomial regression of the previous exercise with the addition of a regularizer (bias). With the use of *bias* λ We tried to regularize my polynomial: basically by summing λ We simplify our result while dumping it using a more complex function for regression, so modifying λ is conceptually similar to change the degree of the polynomial, offering, however, a greater granularity in the choice of the optimal value. λ represents the level of confidence in the quality of the input data: the more the input is noisier, it is better to use λ large and then adopt simple solutions to not fit the noise, the less it is noisy and the more we can decrease λ and use more complex solutions in order to fit data.



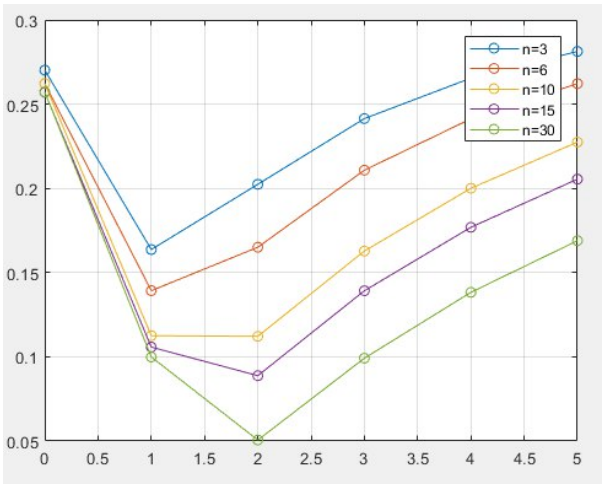
	p = 0	p = 1	p = 2	p = 3	p = 4	p = 5
n = 3	3.808e-01	3.289e-01	3.291e-01	3.302e-01	3.311e-01	3.316e-01
n = 6	3.908e-01	3.236e-01	3.257e-01	3.275e-01	3.287e-01	3.297e-01
n = 10	3.946e-01	3.247e-01	3.293e-01	3.317e-01	3.327e-01	3.332e-01
n = 15	4.596e-01	3.577e-01	3.432e-01	3.379e-01	3.356e-01	3.345e-01
n = 30	5.026e-01	3.201e-01	3.172e-01	3.239e-01	3.275e-01	3.295e-01

(a) fig:lambda = 100 sigma = 10



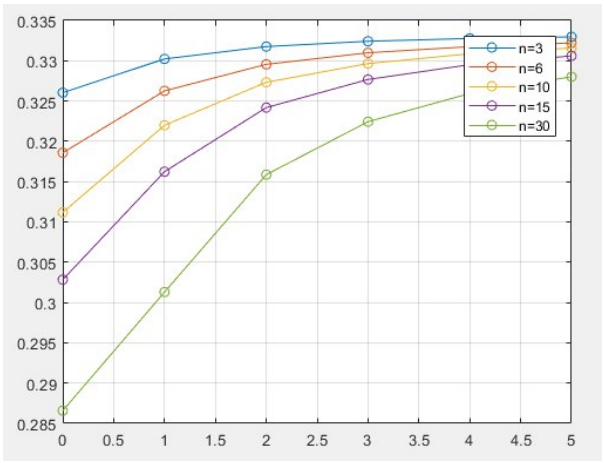
	p = 0	p = 1	p = 2	p = 3	p = 4	p = 5
n = 3	4.252e+00	4.928e+00	5.535e+00	7.421e+00	1.077e+01	1.469e+01
n = 6	2.899e+00	2.286e+00	2.266e+00	2.379e+00	2.562e+00	2.845e+00
n = 10	2.889e+00	2.638e+00	2.204e+00	1.997e+00	1.888e+00	1.902e+00
n = 15	2.672e+00	2.068e+00	1.789e+00	1.636e+00	1.572e+00	1.593e+00
n = 30	1.145e+00	1.117e+00	1.024e+00	9.576e-01	9.037e-01	8.603e-01

(b) fig:lambda = 0,001 sigma = 10



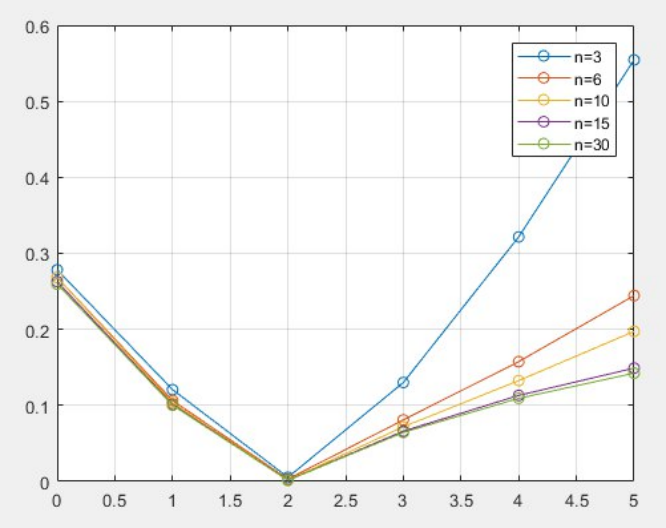
	p = 0	p = 1	p = 2	p = 3	p = 4	p = 5
n = 3	2.702e-01	1.637e-01	2.024e-01	2.414e-01	2.655e-01	2.813e-01
n = 6	2.623e-01	1.393e-01	1.651e-01	2.108e-01	2.413e-01	2.622e-01
n = 10	2.623e-01	1.125e-01	1.122e-01	1.628e-01	2.001e-01	2.275e-01
n = 15	2.571e-01	1.058e-01	8.881e-02	1.392e-01	1.769e-01	2.055e-01
n = 30	2.573e-01	9.990e-02	5.058e-02	9.918e-02	1.383e-01	1.689e-01

(a) fig:lambda = 100 sigma = 10



	p = 0	p = 1	p = 2	p = 3	p = 4	p = 5
n = 3	3.260e-01	3.302e-01	3.318e-01	3.324e-01	3.328e-01	3.330e-01
n = 6	3.186e-01	3.263e-01	3.296e-01	3.310e-01	3.317e-01	3.322e-01
n = 10	3.112e-01	3.220e-01	3.273e-01	3.296e-01	3.309e-01	3.316e-01
n = 15	3.028e-01	3.162e-01	3.242e-01	3.277e-01	3.295e-01	3.306e-01
n = 30	2.866e-01	3.013e-01	3.159e-01	3.224e-01	3.259e-01	3.280e-01

(b) fig:lambda = 0,001 sigma = 10



	p = 0	p = 1	p = 2	p = 3	p = 4	p = 5
n = 3	2.783e-01	1.207e-01	5.476e-03	1.301e-01	3.217e-01	5.545e-01
n = 6	2.682e-01	1.064e-01	3.047e-03	8.095e-02	1.576e-01	2.444e-01
n = 10	2.681e-01	1.036e-01	1.560e-03	7.190e-02	1.326e-01	1.974e-01
n = 15	2.628e-01	1.013e-01	1.481e-03	6.620e-02	1.131e-01	1.491e-01
n = 30	2.596e-01	1.002e-01	1.418e-03	6.451e-02	1.093e-01	1.424e-01

Figure 4: $\lambda = 0{,}00001$ $\sigma = 0{,}01$

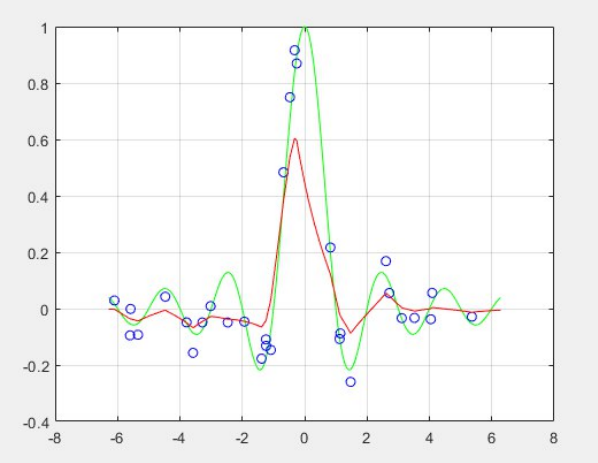
Third exercise

Kernel methods

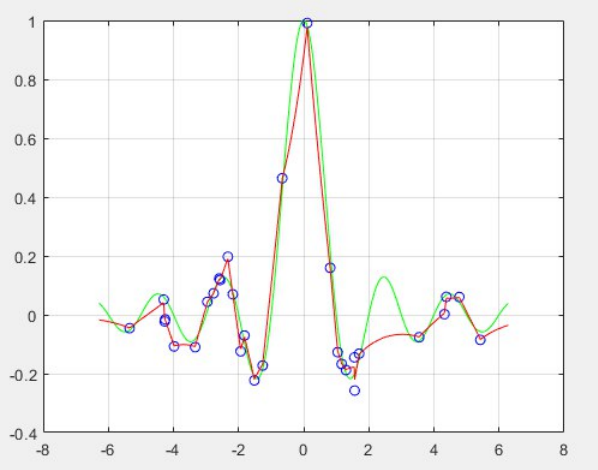
In the multidimensional case we are limited to polynomial type regressions due to the curse of dimensionality, so we use kernel methods, in particular the Gaussian kernel.
Use two hyperparameters:

- λ that regularize the solution $\rightarrow$ the higher it is, the more the model is flat
- the 3 hyperparameters of the kernel, in this case only the gamma is taken into account: a parameter that regulates the no linearity of the solution

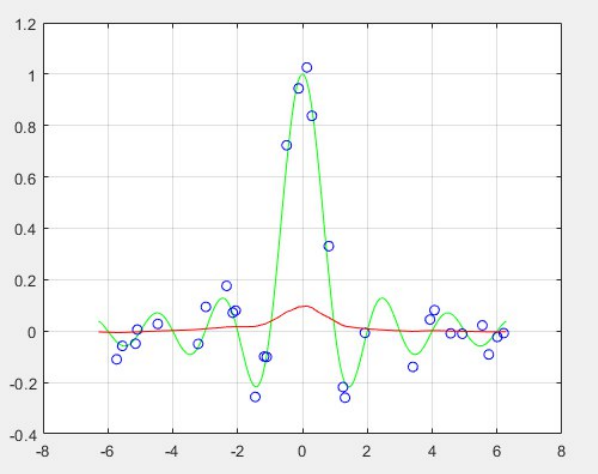
there are combinations of λ and gamma that more or less lead to the same solution. So λ and γ are parameters that are not completely disconnected, you can find more or less the usual models with different values of these quantities.
How to find the best λ and γ values:
We find best values of hyperparmeters through validation procedure (split the data, create the model with part of the data, and test the quality of hyperparmeters on a set of data that are not used during the model creation), the best combination of hypers is the one that minimize the error on the validation set($\rightarrow$ the set of data not used for creating the model) The best thing to do is to mediate the results.
The splitting of data should not be done only once, but should be repeated because with a single report the variance is higher, but if you take the average I'm more confident that my estimator is close to the real average.
Observation: the loop on k is put as the first loop in our implementation (putting it inside I could avoid keeping the array of all errors) because my estimate with k inside would be subject to 2 types of variance: a variance due to the estimator same that can not be eliminated, the other is the variance of how the learning set and the validation set occur. Risk of selecting bad λ and γ because my solution is determined by the variance of the quality of the splitting. Writing the code as we wrote it, the splitting part is always the same for every γ and λ and this reduces the variance of my estimator.
Last observation: The graph, even increasing range, does not pass from all points because of the 64-bit precision that is not able to reach them all.



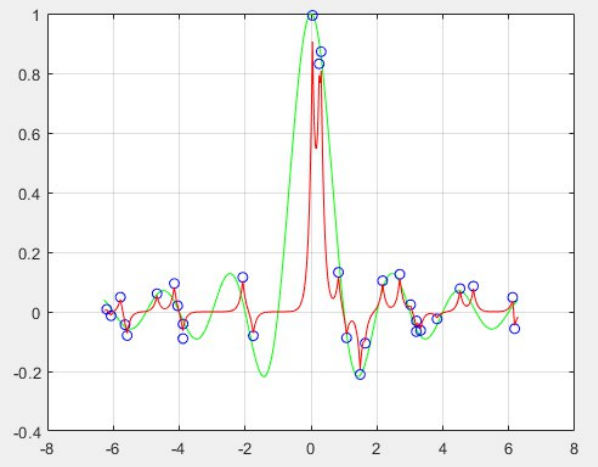
(a) fig:$\lambda = 1$ $\gamma = 1$



(b) fig:$\lambda = 0{,}01$ $\gamma = 1$ Less smoothness



(a) fig:$\lambda = 25$ $\gamma = 1$ More smoothness



(b) fig:$\lambda = 0{,}1$ $\gamma = 10$ More non linearity

Very irregular and non-linear, it can not reach all the points anyway due to the precision of the machine.

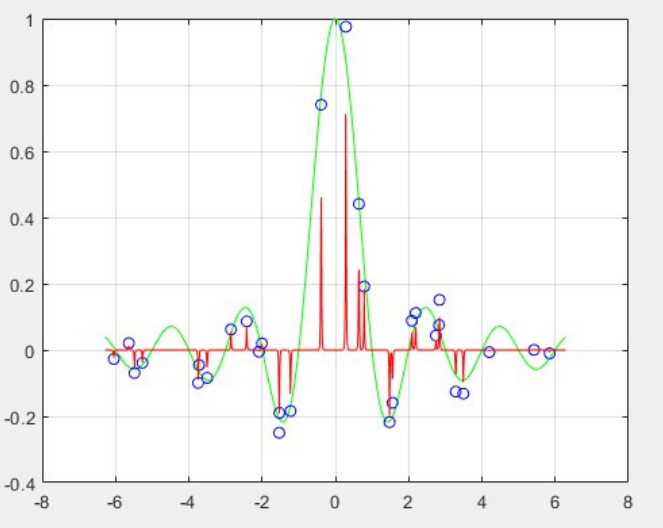


Figure 5: $\lambda = 0$ $\gamma = 100$

Real World Problem

We face a real-world problem as the last lab.

After understanding the problem and understanding what method use to solve it, we proceed by checking data in the dataset:

- Categorical features
- Missing values
- Numerical problems

We normalize the data so that each feature has the same weight in the solution. We control the distribution of data, checking the various categorical features that possibly have a Gaussian distribution: the Gaussian distribution indicates that data come from nature. We observe that during the calculation two types of errors are kept:

- Error on validation: used to optimize γ and λ
- Error on the test: tells me that if we take the data and we do learning with those particular γ and λ, we get a model with that particular unbiased error

To verify the goodness of model a single parameter is not enough (error for example), then we print a scatter plot: x axis $\rightarrow$ true value, y axis $\rightarrow$ prediction.
Ideal case: there is a line, so the values coincide exactly.
Real case: there is a bubble, an elipsoid is formed around the straight line Using more data for learning we can improve prediction quality, so we can try to change percentages.

We note that $\lambda_{best} = 10^{-4}$ and $\gamma_{best} = 2.2122$.
λ indicates the smoothness of the model and γ indicates its non-linearity, but in itself we can not evaluate whether they are large or small. To evaluate whether the γ is large or small we have to evaluate the dimensionality of the space, if we are in a low-dimensional space, $\gamma = 2$ is probably large, while in one of great dimensionality it is probably small. In turn, lambda depends on the cardinality of the space, it happens that if we have small numbers in ω, λ will be large and then very regularize the solution. In this case the size of the space is 11, so $\gamma = 2$ can be considered small.

We verify the distribution of the data, here for example the first categorical feature (X (:, 1)) the fact that it has a Gaussian distribution is a good sign because it is the correct distribution for data coming from nature:

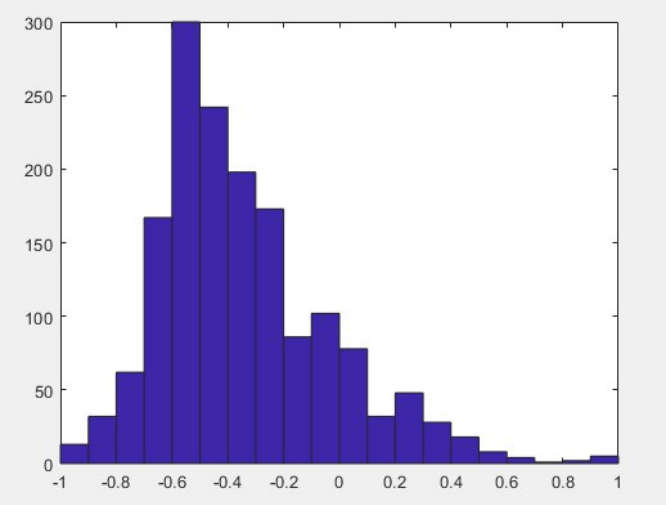


Figure 6: $\lambda = 1$ $\gamma = 1$

Solution with γ_{best} and λ_{best}, values that minimize the error

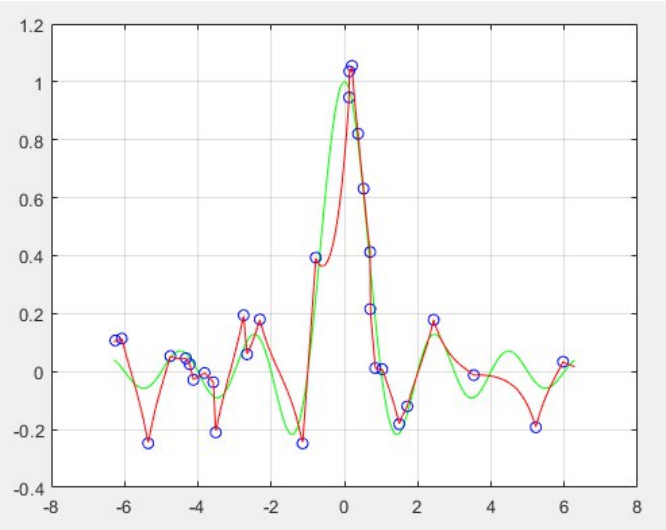


Figure 7: λbest γbest

We verify the distribution of data, here for example the first categorical feature (X (:, 1)) the fact that it has a Gaussian distribution is a good sign because it is the correct distribution for data coming from nature

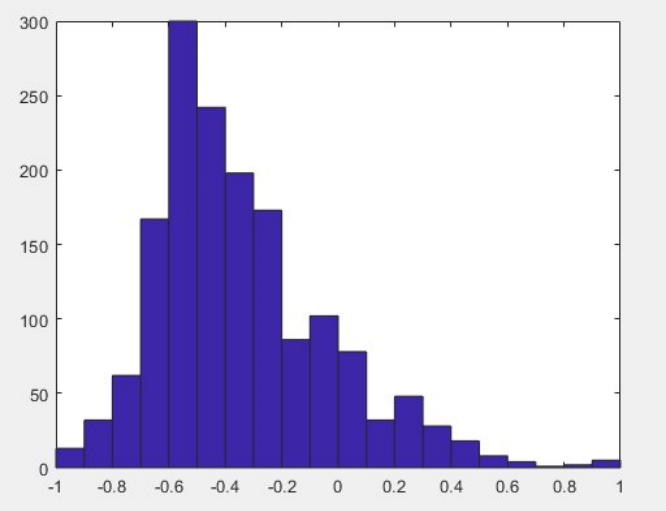


Figure 8: λbest γbest

Scatter plot: x axis $\rightarrow$ true value, y axis $\rightarrow$ prediction

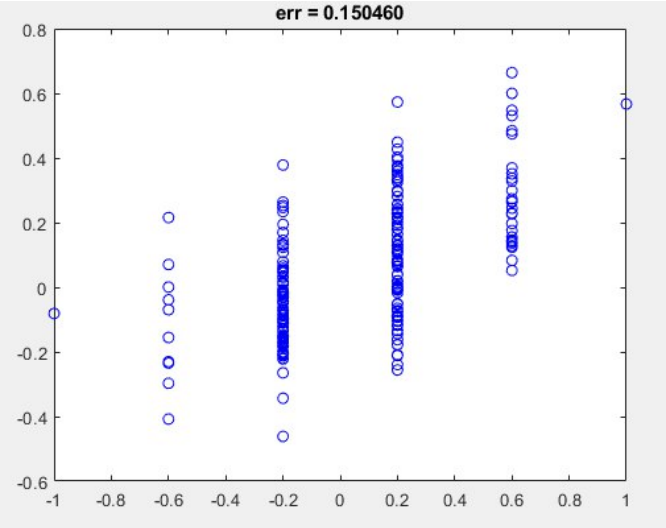


Figure 9: λbest γbest

Recalculated with k = 30

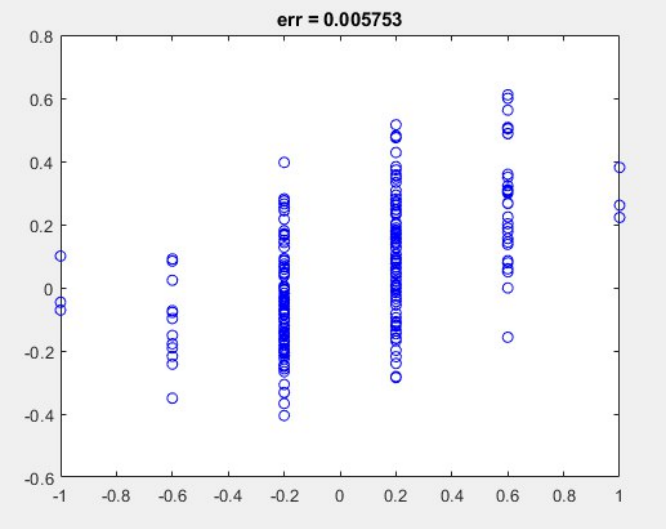


Figure 10: λbest γbest

List of Figures

1 Regression Function 2

2 Comparing error with different values of p 3

3 Comparing error with different values of p 3

4 $\lambda = 0{,}00001$ $\sigma = 0{,}01$ 5

5 $\lambda = 0$ $\gamma = 100$ 7

6 $\lambda = 1$ $\gamma = 1$ 9

7 λbest γbest 9

8 λbest γbest 10

9 λbest γbest 10

10 λbest γbest 11