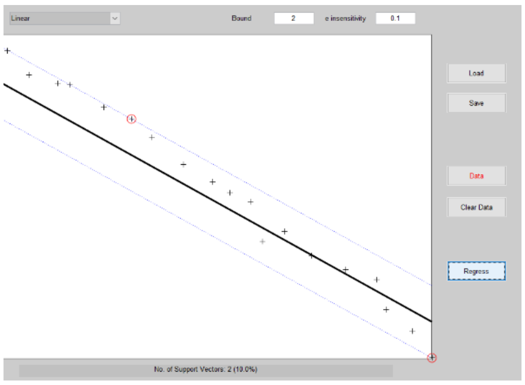
Kenneth Devloo, s0219469

Support vector machines: Exercise session 2

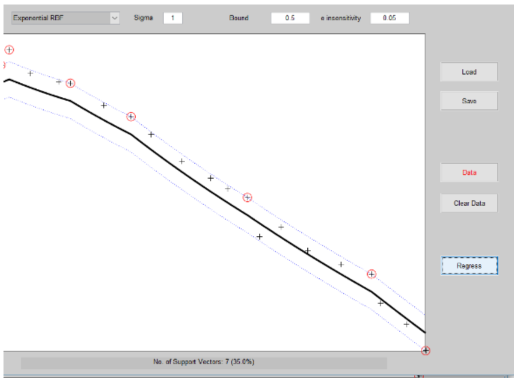
2.1 The Support Vector Machine for Regression

The value of e is the desired accuracy. The higher it is the more error you allow. The bound is the regularisation parameter. The linear kernel is better for linear functions. It requires less support vectors. (See first image). Given s also an approximation with and RBF kernel (Second image).

In case of the linear kernel, the value for e is the allowed margin. A high value of e will result in a horizontal estimation since there is too much flexibility allowed. A higher bound will take more support vectors into account in the final solution. Sparsity comes in when a solution with only a few support vectors is found.

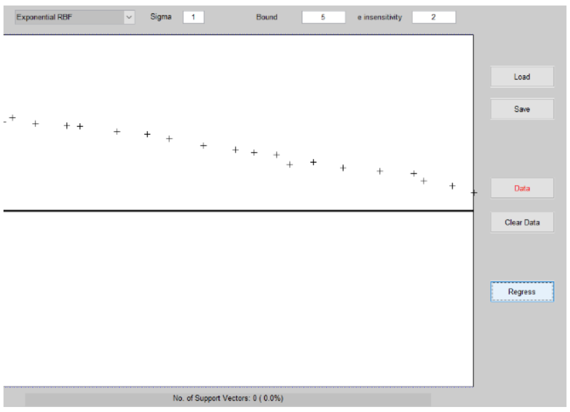

Linear   Bound 2   e insensitivity 0.1

No. of Support Vectors: 2 (10.0%)

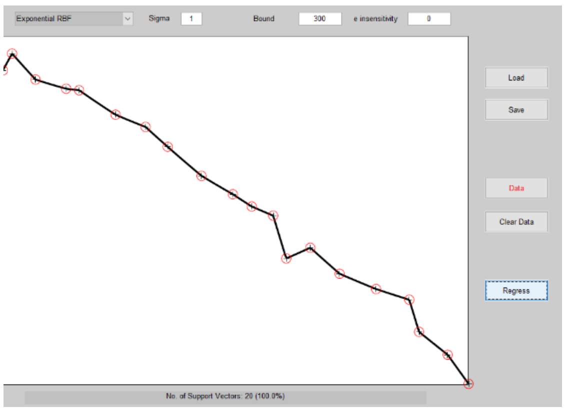

Exponential RBF   Sigma 1   Bound 0.5   e insensitivity 0.05

No. of Support Vectors: 7 (35.0%)

In the polynomial case. We will not approximate with a straight line. It is easier trying to overfit (Second image). Changing the bound (regularisation) will help to get a better fit. We should mind the e parameter again because if it is too high, we will not approximate the correct line. As it won't matter much anymore, even with regularisation. Some images below to illustrate these things. On the left: High value of e and high bound. No support vectors taken into account. On the right: Overfitted line with low e and high bound, that also has no sparsity at all due to the use of all support vectors.


Exponential RBF   Sigma 1   Bound 5   e insensitivity 2

No. of Support Vectors: 0 (0.0%)


Exponential RBF   Sigma 1   Bound 300   e insensitivity 0

No. of Support Vectors: 20 (100.0%)

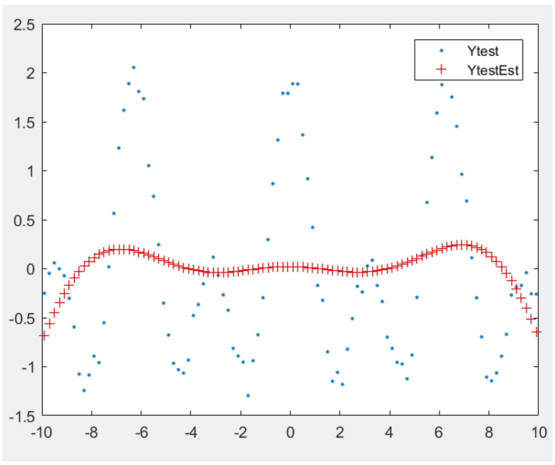
Do note that if that function was not linear, the approximation of the RBF kernel could actually be the more precise one.

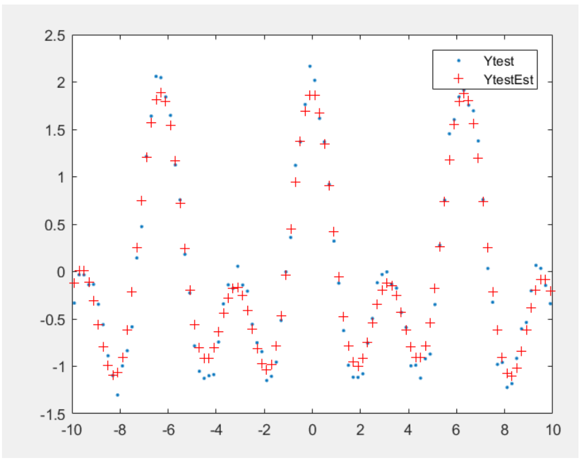
2.2 A Simple Example: Sum of Cosines

The obtained approximation with the given code is underfitted as shown in the first image on the left.

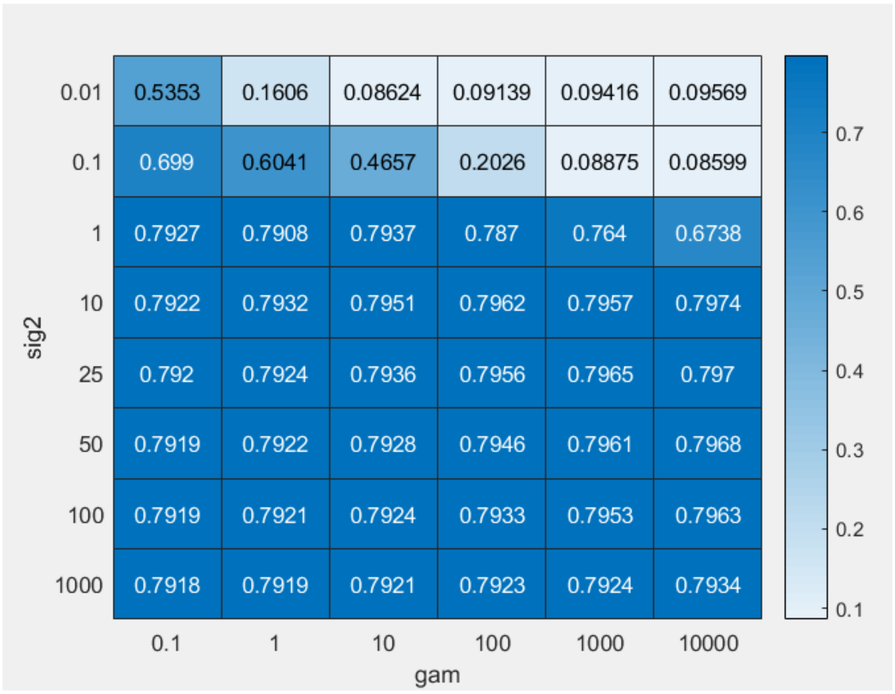
Lowering sigma will result in a less straight here since high models with high sigma approximate linear solutions. It was lowered to 0.1. Raising the gamma parameter also results in a better fit. It was raised to 200. The result is on the right.

The mean squared error goes from 0,875 to 0,023 that way.





Also shown is a heat map showing the MSE for different values of gamma and sig2. From the tested values, gam = 10000 and sig2 = 0,1 seems to do best here.



sig2 \ gam	0.1	1	10	100	1000	10000
0.01	0.5353	0.1606	0.08624	0.09139	0.09416	0.09569
0.1	0.699	0.6041	0.4657	0.2026	0.08875	0.08599
1	0.7927	0.7908	0.7937	0.787	0.764	0.6738
10	0.7922	0.7932	0.7951	0.7962	0.7957	0.7974
25	0.792	0.7924	0.7936	0.7956	0.7965	0.797
50	0.7919	0.7922	0.7928	0.7946	0.7961	0.7968
100	0.7919	0.7921	0.7924	0.7933	0.7953	0.7963
1000	0.7918	0.7919	0.7921	0.7923	0.7924	0.7934

2.3 Hyper-parameter Tuning

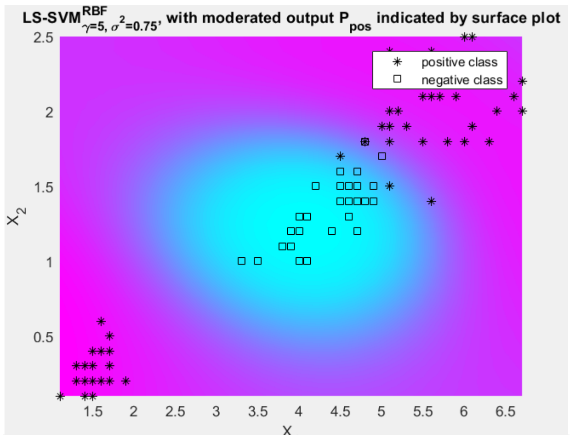
Just like in the first exercise session, I did multiple runs of tunelssvm to obtain better parameters. The global optimization functions for this implementation are not fully documented. From what I can find, randomized directional search is a gradient descent algorithm with multiple starting points. Using directional search multiple times will also cause a crash but it is apparently not well supported.

Grid search is just an exhaustive search hat will require more time then simplex that looks for a more optimal direction and thus requires less calculation. Grid search also seems to generate values with slightly higher costs.

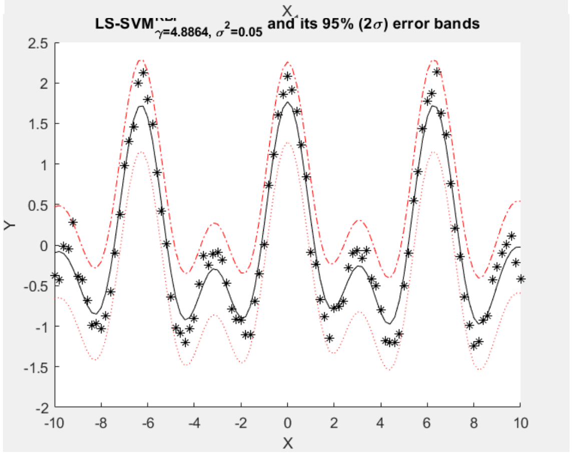
All algorithms perform reasonably well. Values for gam still have big differences. It would probably be more interesting to see differences on a more complex dataset.

2.4 Application of the Bayesian Framework

In the Bayesian framework, each level builds upon the assumption that higher levels are known. For level 1, it is assumed that the hyperparameters and kernel are known and that is the evidence provided for this. The same connection exists between level 2 and 3. The functionality gives us the most optimal parameters that can be used.

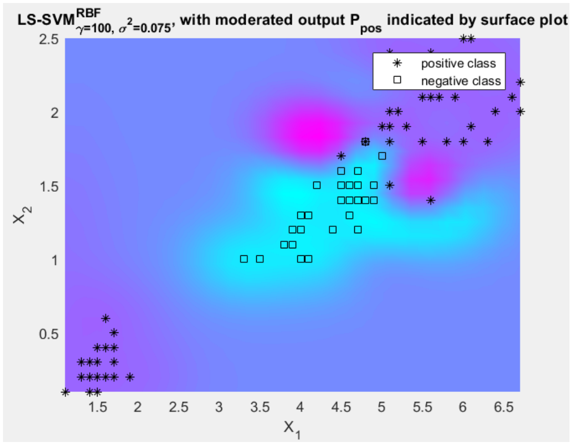


LS-SVM now has a distribution of w and b that can for example help generate regression results. That is the chance that for a certain X, we obtain a value Y, given the current model determined with the data. This chance is illustrated in the first picture below. The distribution for Y on each input value X is shown along with the error bands. It can be seen as the uncertainty that a certain point belongs to a certain class.



In the second picture, this distribution is shown for the classification problem. This time, the (un)certainty of an input belonging to a class is shown with a colour. One way to imagine this is with a Gaussian distribution in the vertical direction on every point.

Some tests with different sig2 and gam values have also been done. In general, that gives a similar result as with normal classification like discussed in the previous report. One thing that is visible is that choosing extremely high values for gam will give the negative class more weight. Extremely low sig2's will make this effect even stronger as illustrated below. High values of sig2 will ruin the results in the sense that the negative class will dominate the entire plot.



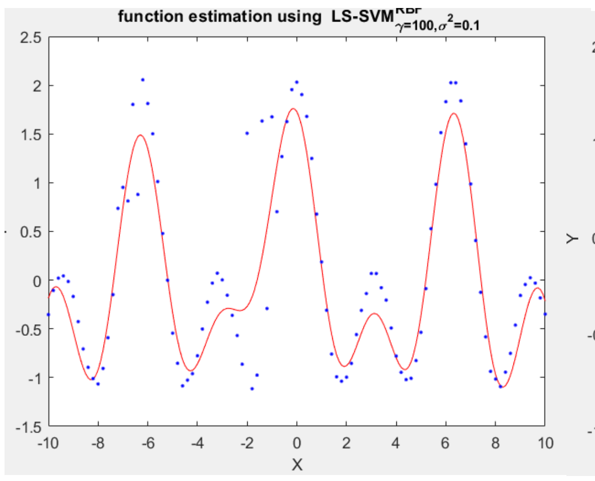
When using the framework for automatic relevance detection, we look a good value of sig2 to test different parts of the input vector. The input dimensions are ranked according to their relevance. If possible, dimensions can be removed to optimise performance.

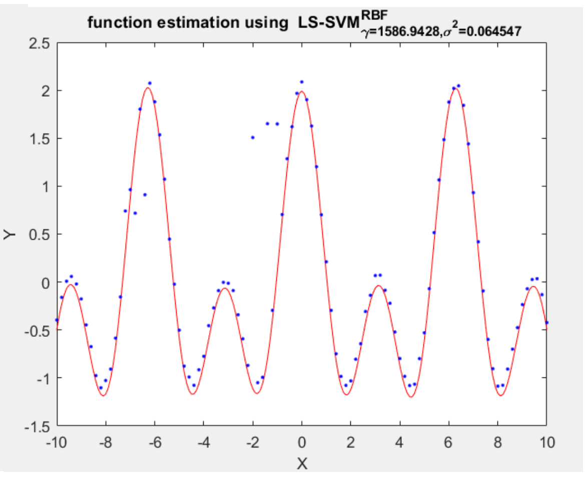
You could reach something similar with cross validation or 'leave one out' by trying different sample dimensions instead of input variables and see how that changes the error rate.

2.5 Robust Regression

In robust regression, we want outliers to have less influence on the result of the cost function. That is why it is preferred not to square the distance like with the mean squared error, but to use for example, the Manhattan distance.

If we don't use more robust regression, outliers will have too much weight and therefore approximation can be wrong at some points (left). On the right is a result coming from a weighted cost function.



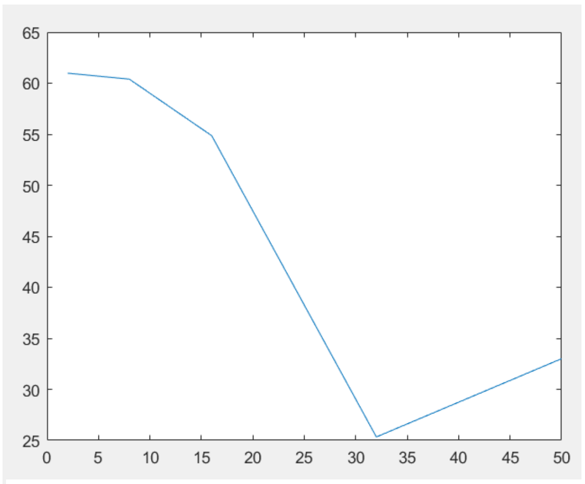


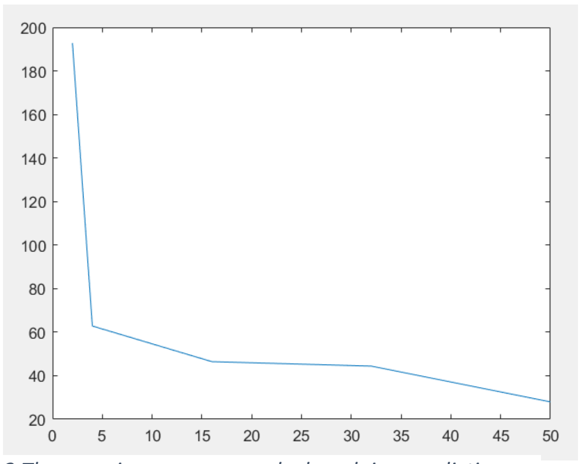
2.6 Homework Problem

In this part, I was at first, I spent too much time working with the old version of ls-svmlab.

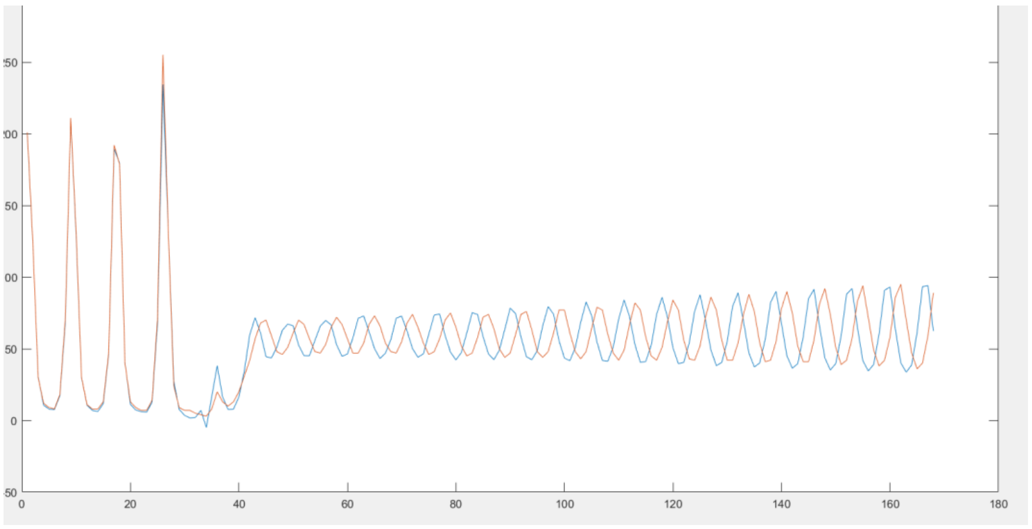
I have used the santa fe dataset to look how the order affects performance. I do think that using the order as 50 would be a decent value if it does not take away too much test samples. Being able to use a lot of history is handy to capture more variation. But if we need to train for so many features, we need enough samples. A good balance has to be found here. In this specific example, I tested different values for this and an order of about 32 seems most effective. I only used one validation set for this but the idea is clear that we could use n-fold cross validation.

Image 1 on the right shows the square roots of the MSE's for the chosen values of order. Doing this for the same points in time gives a very different result. In the 2nd image, this error is only done for values 51-200 from the test set as I wanted to compare for the same points in time. We have to be careful when using the MSE since errors can also occur in timing as is shown in the 3rd picture. This is mostly true when trying to predict far away in time. Therefore, the second graph with the errors only shows the error on the first 50 time points to limit this effect. A more sophisticated algorithm taking in account this time error would be more recommended. But it is out of scope for this homework.


1 Comparing the errors with an as big a possible amount from the test set. Order.32 seems like the winner. But it could produce values earlier in the test series!


2 The error is now compared when doing predictions starting at point 51, ending at point 100. The result for order = 8 was left out is at had an outlier preducted whuch messed up the results. The higher order '50' is the better one here.

3 Time series predicted on a large time range. We see that the biggest error is actually in time.



Using the same sig2 and gam for validation and prediction seems like a good idea here since it is a normally trained svm that is used here. Prediction does nothing more than regression using some input values. In this case this is the previous amount of values of which the amount is equal to the order.