Support Vector Classification
Session 2

Tijmen Wintjes

May 14, 2017

1 Function Estimation

1.1 Support Vector Machine for regression

The twenty data-points I created are displayed below.

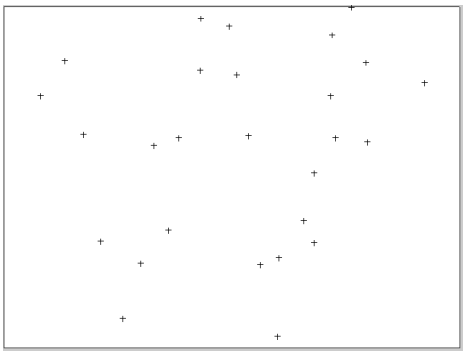

Figure 1: The data generated for the 'uiregress' function.

All kernels available in 'uiregress' funciton of the SVM-toolbox were applied. The results for the linear, linear spline and rbf-kernel are displayed below. The RBF-kernel performs best.

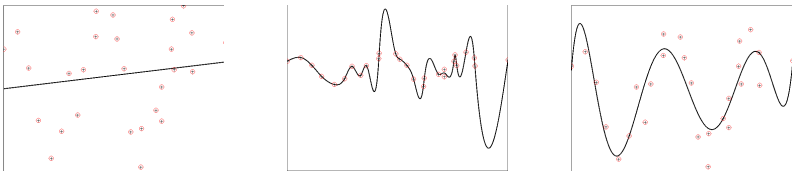

Figure 2: Estimations of the trigonometric function with various kernels.

Next we investigate the effect of the hyperparameter e. It gives the band around the estimated function for which data points lying inside have been

estimated perfectly. Increasing it makes the estimation less sensitive to noise, but also less precise.

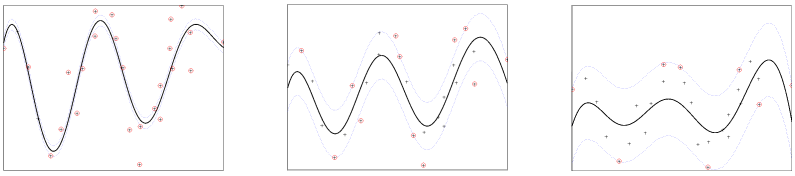

Figure 3: The effect of the hyperparameter e, from left to right $e = 0.01$, $e = 0.05$ and $e = 0.1$. $\sigma = 2$.

The bound parameter defines the penalty on misclassification. If this value is very low. The estimation is not sensitive enough. Setting it too high makes the estimation sensitive to outliers.

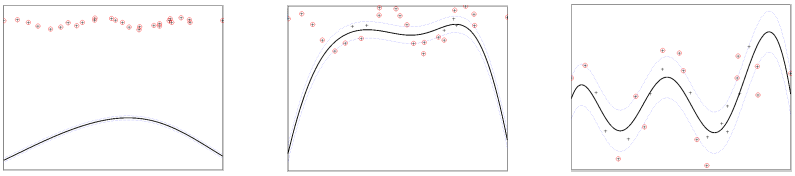

Figure 4: The effect of the hyperparameter $Bound$, from left to right $Bound = 0.1$, $Bound = 1$ and $Bound = 10$. $\sigma = 2$.

In figure 3 we can nicely see that increasing e leads to sparsity of the vector with support vector values. SVM is different form least-square function estimation in the metric it uses to minimize. As long as points are within the e-band they are treated as if they lie exactly on the line itself and their distance has cost $= 0$.

1.2 A simple example: The *sinc*

For a first exploration of the function estimation abilities of the lssvm we use a noisy sinc function.

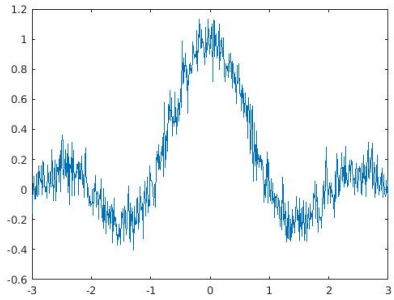

Figure 5: Plot of the noisy sinc function

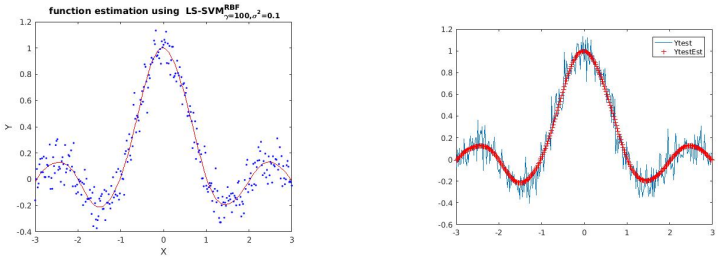

Figure 6: On the left a plot of the estimated function using an lssvm with rbf-kernel, $\sigma = 1$ and $\gamma = 100$, on the right a comparison with the test data.

1.3 Hyper-parameter tuning

Tuning of the hyper-parameters σ and γ was carried out with both a gridsearch and simplex optimization scheme. The difference between the two being that gridsearch is a brutforce method, while simplex uses an algorithm devised by Nedler and Mead to iteratively find values of σ and γ which lead to lower costs. The optima found are however not unique, rerunning 'tunelssvm' multiple times leads to different results. However the final value of the objective function stays level with respect to the sought accuracy. Qualitatively there seems to be little difference between the two methods.

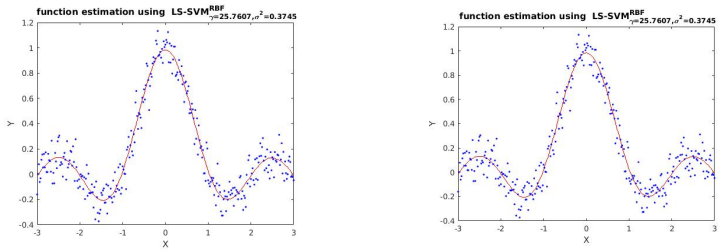

Figure 7: On the left a plot of the estimated function using 'gridsearch' to find σ and γ, on the right we used 'simplex'

1.4 Application of the Bayesian Framework

Another way to tune hyper-parameters is by using the bayesian interference framework for a lssvm. Page 120 of the course notes gives a decent graphical overview of how conceptually the levels of interference are related. Interference for the function estimation problem of the sinc function leads to the following result.

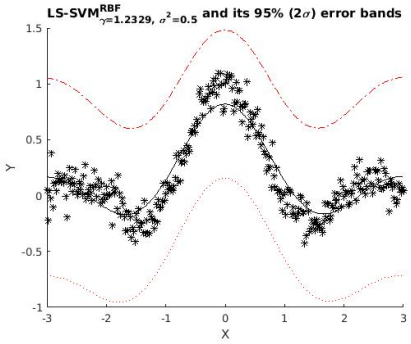

Figure 8: Plot of the lssvm with hyper-parameters infered by the bayesian framework

Bayesian inference also allows us to define a posterior class probability in classification problems. For the iris data set such a approach was taken in the figure below. The figures show that choices of the hyper-parameters that are not optimal result in a blurring of the decision boundary. Left a picture of the posterior class probability for the $\gamma = 0.5$ and $\sigma = 0.75$, the left two pictures are made with in the middle $\gamma = 10$ and on the right $\sigma = 10$. The colors give us the probability that a data point within that area belongs to a class.

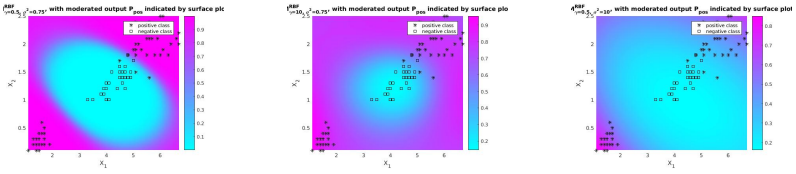

Figure 9: The effect of the hyperparameter γ and σ on the decission boundary of the iris data-set.

The bayesian network can also be used for Automatic Relevance Detection, which means trying to to couple an output to inputs which in some metric seems to be best corresponding. Coupling the randomized input of the sinc to output we get the following figures:

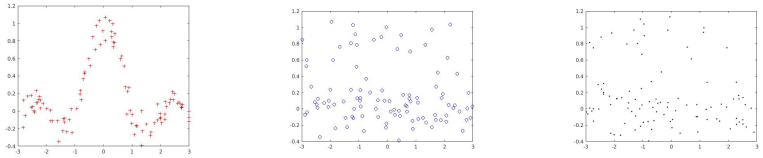

Figure 10: Three different randomized x-value inputs and the corresponding sinc output for one of them.

The bayesian ARD function suggest that the first input is probably the input that generated the data. A similair result can be obtained by performing crossvalidation with an RBF-kernel. For the different plots we sort the cost returned by the crossvalidation and rank them from low to high cost. This will be the corresponding ranking.

1.5 Robustness

This subsection discusses the robust methods of the lssvm toolbox. Adding a couple of outlier to the noisy sinc function and estimating it a normal way makes clear that outliers have a big effect on the least square method.

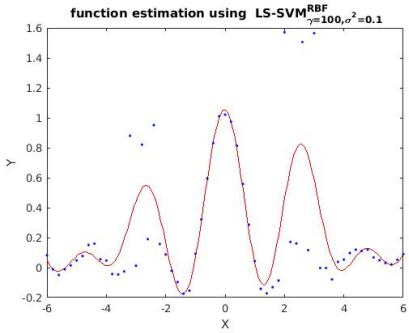

Figure 11: Plot of the estimated noisy sinc with outliers

For the robust case we prefer the mean absolute error over the mean squared error. The squared error amplifies the cost of having data points far away from the estimated curve, which in this case we don't mind, therefore we take the mean absolute error which is less sensitive. Page 152/153 give a clear overview. We fit the sinc function with four different loss functions. The huber and logistic function have a ceiling for the cost of the strong outliers, but still count the. The Hampel and Myriad function barely take them into consideration at all.

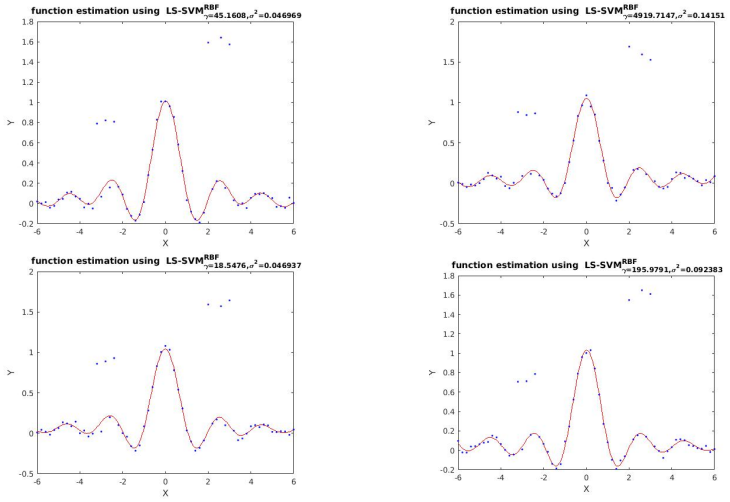

Figure 12: Four different robust estimations with differnt cost function. Top left Huber, top right hampel, bottom left logistic, bottom right myriad.