Support Vector Machines: Methtods & Applications
Assignment 2: Function Estimation & Time-Series Prediction

Thibault Putseys

May 25, 2018

1. The Support Vector Machine for Regression

2. A Simple Example: Sum of Cosines

In this exercise, we are asked to investigate the impact of hyper-parameters on an SVM regression model for modelling a sum of cosines. Using an RBF-kernel, we apply a grid search for the kernel parameter σ^2 and the regularization parameter γ. The optimal tuning parameters are found for $\gamma = 2050, \sigma^2 = 0.1$, with $MSE = 0.01$. Note that MSE does not vary significantly with γ for low values of σ, id est lots of equally good models exist. The fit of the final model is shown in 2.

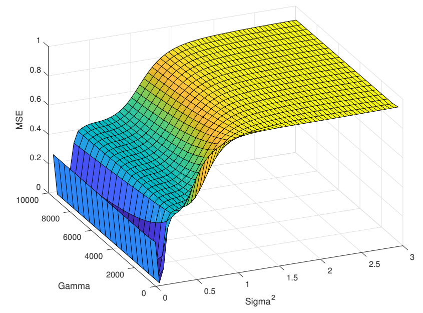


Figure 1: Sum of cosines: grid search for γ, σ^2.

3 Hyper-parameter tuning

The LS-SVM Toolbox has some nice built-in functions which allow to tune hyper-parameters automatically. These algorithms work in two stages. First, the starting values are calculated using a global search algorithm. Thereafter, a local search algorithm is used to refine. In order to check the performance of these methods, several performance indicators are logged while running the sum of cosines example. Every method has been ran 20 times repeatedly in order to check the stability of the results and the median meta-parameters and MSE are shown in Table 3. Please note that runtime is expressed as a total runtime over 20 successive runs. Total runtime indicates that simplex is a faster local search algorithm as compared to grid search, which makes sense intuitively since it avoids searching the whole grid. On the other hand, grid search is more reliable in terms of stability. An example can be seen by looking at the median parameter settings for σ^2 for a combination of simplex and simulated annealing. It is clear that a bad solution as this one can lead to a sharp increase in MSE. Secondly, we can see that randomized directional search takes longer than simulated annealing. This makes sense, since randomized directional search requires several random restarts, but is therefore more trustworthy in terms of delivering a proper starting position for the local search algorithm in the second phase.

4. Applications of the Bayesian Framework

In a Bayesian setting, three levels of inference can be distinguished. The highest level can be

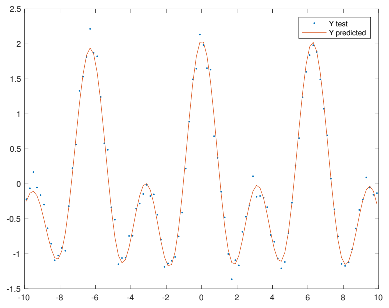
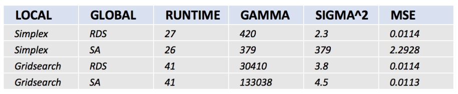


Figure 2: Sum of cosines: line fit.



LOCAL	GLOBAL	RUNTIME	GAMMA	SIGMA^2	MSE
Simplex	*RDS*	*27*	*420*	*2.3*	*0.0114*
Simplex	*SA*	*26*	*379*	*379*	*2.2928*
Gridsearch	*RDS*	*41*	*30410*	*3.8*	*0.0114*
Gridsearch	*SA*	*41*	*133038*	*4.5*	*0.0113*

Figure 3: Performance of LS-SVM tuning algorithms.

used for model selection purposes as it represents the probability of the model given the data, the second level for the tuning of hyperparameters whereas the lowest level is used to parametrize model weights corresponding to each input variable.

Another nice thing about the Bayesian framework, is that a full distribution of estimates is obtained as compared to a point estimate, which makes it easy to add confidence bounds to estimates made. These confidence bounds can be used in a backwards model selection procedure called Automatic Relevance Determination. In this case, one starts with building a full model including all covariates and iteratively removes the covariate with the lowest probability of having a positive weight in the model, compares if removing the covariate adds value by comparing the posterior probability of both models and continues until convergence. In the exercise, 2 redundant variables are added to the sum of cosines example and it is checked that the retained model only includes the original variable. The selected variables are 1 and 3, meaning, the 3rd variable is wrongfully selected by the procedure. After all, this is still an algorithm prone to error and statistical variation. Another way of doing backwards selection could be to iteratively remove one variable based on the sharpest decrease in cross-validation error until there is no further decrease. Also here, there is no guarantee of finding the best model in the end.

Another nice application of using priors in a Bayesian setting, works in for the case of classification. In case we introduce a prior assumption on the probability of a datapoint belonging to a certain class and priors on hyperparameters taking on certain values, we can calculate the probability at any prediction point. In the example of the exercise session we see that the parameters in the conditioning set influence the outcome of these probabilities significantly. All else equal, a larger regularization parameter causes probability estimates to be pushed more towards the a priori estimate, which is 50% in case of no information. In Figures 4 and 5, we compare a gamma parameter of 10 versus 100. In order to see clearly the effect, please note the different scale used in both plots on the right hand side of the graph. Secondly, we can see that increasing the bandwidth parameter σ decreases the surface that will be considered a positive prediction and vice versa, since the used kernel function is less peaked. Note that this behavior is exactly similar as in previous assignment, although the introduction of a specific prior other than the uninformed one could alter results. Classically, in case enough data is available, the effect of the prior on the actual prediction loses importance.

The Bayesian framework can also be applied as a backwards model selection procedure using Automatic Relevance Determination.

5. Robust regression

Outliers can have a significant impact on the predictive performance of statistical models. Robust estimators try to limit the impact of outliers on the estimation procedure by using absolute er-

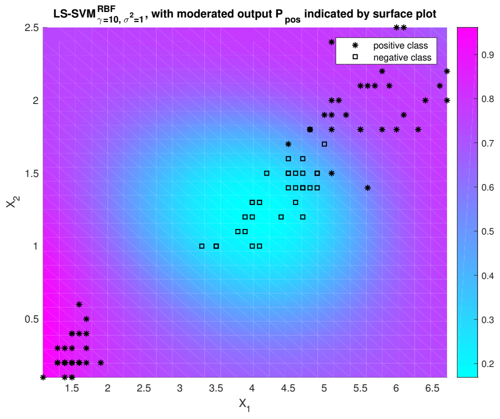


Figure 4: Bayesian estimation: $\gamma = 10$

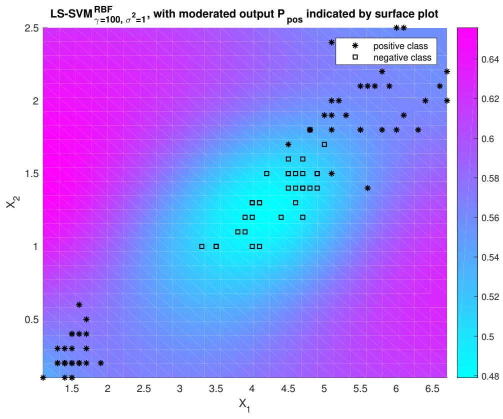


Figure 5: Bayesian estimation: $\gamma = 100$

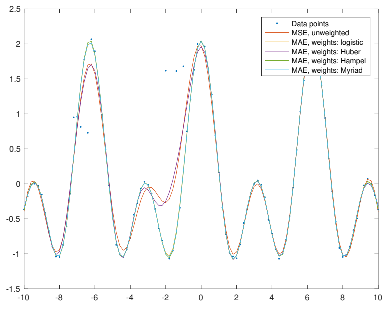


Figure 6: Predictive performance of robust estimators

rors instead of squared errors in the estimation together with a weighting scheme. Outliers are added to a toy example and the fit of a regular SVM using an RBF kernel leads to a bad fit as shown in 6. Several robust weighting schemes are tested and compared. We can see from the graph that most robust estimators are able to reconstruct the underlying by outliers as compared to the regular case. Only Huber weights seem to have difficulty finding the underlying curve.

6. Homework Problem

6.1 Logmap dataset

We are asked to try and improve a time series model for a provided dataset consisting of 200 observations in total. For this purpose we split the dataset in a training set of 100 datapoints, a validation set of 50 datapoints and a test set of 50 datapoints. The training and validation set should be used to build a model which is then used to predict the remaining 50 from the test set. The model presented in the exercise session with order 10 is used as a baseline with an error on the test set of 13% MSE. Note that we take the same number of observations in the validation and test set since our goal is to improve our predictive power over that specific timespan.

For selecting the right order of the model, we apply the same procedure for every lag and af-

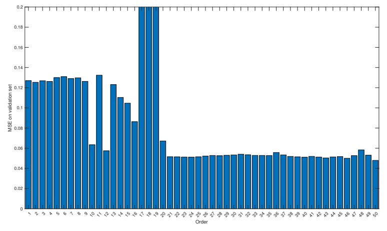


Figure 7: Logmap dataset: selecting the number of lags on a validation set.

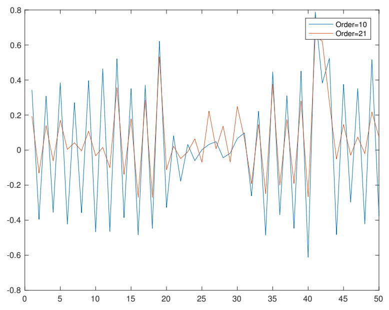


Figure 8: Logmap dataset: predictive performance of initial and final model.

terwards check the performance of the obtained model on the validation set. We look at lags 1 till 50, and for every lag we fit an SVM regression model using an RBF-kernel fitted with 10-fold crossvalidation and simplex as a local optimization algorithm. The results are shown in Figure 7. We can see that after order 21, the results impact of increasing the order seems to stabilize. Following the principle of Occam's Razor, we decide to tak order 21.

The final error of the obtained model equals 5% MSE, which is 8 percentage points below the baseline. The improvent of fit compared to the baseline is shown in Figure 8. Here, the error of both models compared to the true line is plotted. Clearly, the new model is better at nearly all timesteps.

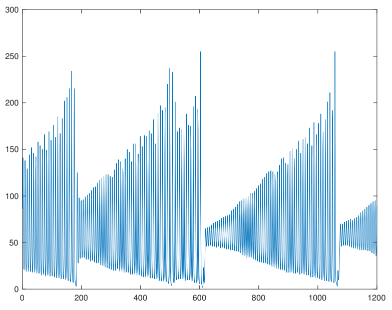


Figure 9: Time series - Santa Fe laser dataset

6.2 Santa Fe dataset

We are asked to develop a time-series model for the Santa-Fe chaotic laser dataset containing 1200 observations. Similarly as for the logmap dataset, we split the data in a training set of 800 observations and equally divide the remaining observations into a test and validation set. The total series is plotted in Figure 9 and shows a way more erratic behavior as compared to the previous series. We apply a similar procedure to find a fit for this task.

For every order in the range of 1 to 200, we fit a model using 10-fold cross-validation and and check the performance on the validation set. In Figure 10 we can see the Mean Squared Error on a logartihmic scale for every lag. We used this scale since we see sharp peaks for certain lags. It doesn't seem to be an easy task to select a proper candidate in this case. Around lags $38 - 40$, there appears to be a drop in MSE. Therefore we take 38 as the final model. The predictive performance of this model on the test set is shown in Figure 9. We can see that it succeeds quite well in fitting the second half of the series, while the peaks in the beginning are more difficult to fit.

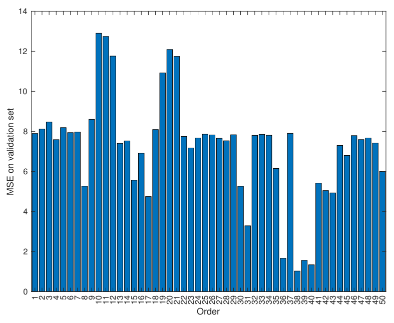


Figure 10: Santa-Fe dataset: selecting the number of lags on a validation set.

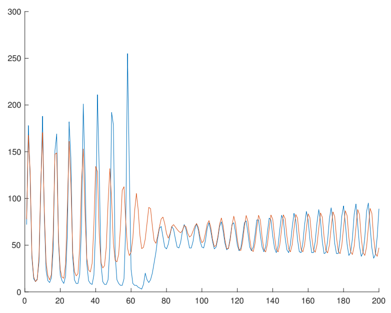


Figure 11: Santa-Fe dataset: Predictive performance of final model with order = 38.