SVM2

Irene Volpe
r0740784

1 Exercises

1.1 The Support Vector Machine for Function Estimation

1.1.1 A dataset where linear kernel is better

Fig 1: a dataset of randomly created points that doesn't have any distinguishable features.
If we take this dataset to be a series of Cartesian coordinates, then we can image there to be a functional relationship between the x and y coordinates of the datapoints.

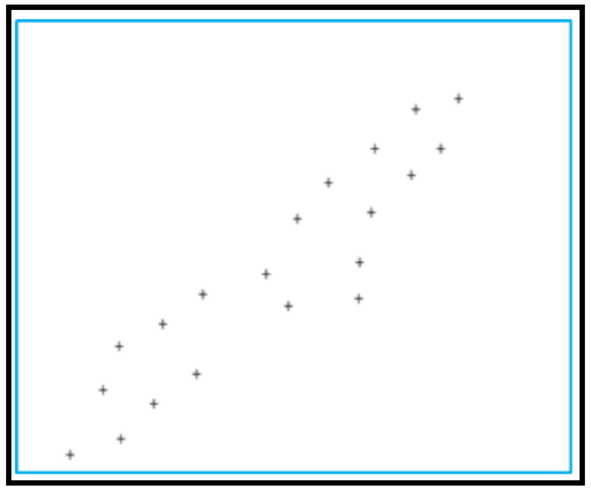


fig 1

1.1.2 Influence of e (width margin, n° of SVs)

We explore relationship between the set of datapoints using a linear kernel.
'e' = insensitivity parameter, determines accuracy margin; controls the width of the insensitive zone.

Fig2:
a) e = 0.1, SVs = 20 —> margins almost nonexistent.
b) e = 1.0, SVs = 20 —> narrow margin
c) e = 4, SVs = 2 —> margins gets wider. BEST MODEL
d) e = 8, SVs = 0 —> margin gets wider and almost horizontal
e) e = 10, SVs = 0 —> margin gets wider and horizontal
f) e= 100, , SVs = 0 —> classification distorted.

SVM goal: find a decision surface maximally far away from any data point.
This distance from the decision surface to the closest data point determines the margin of the classifier.

1) small value of 'e' —> less margins width (ignore errors that are situated within certain distance of the true value)
2) large margin —> avoid misclassification.
3) higher value of 'e' —> less number of SVs leading to data compression.
4) less SVs —> faster classification, less overfitting;

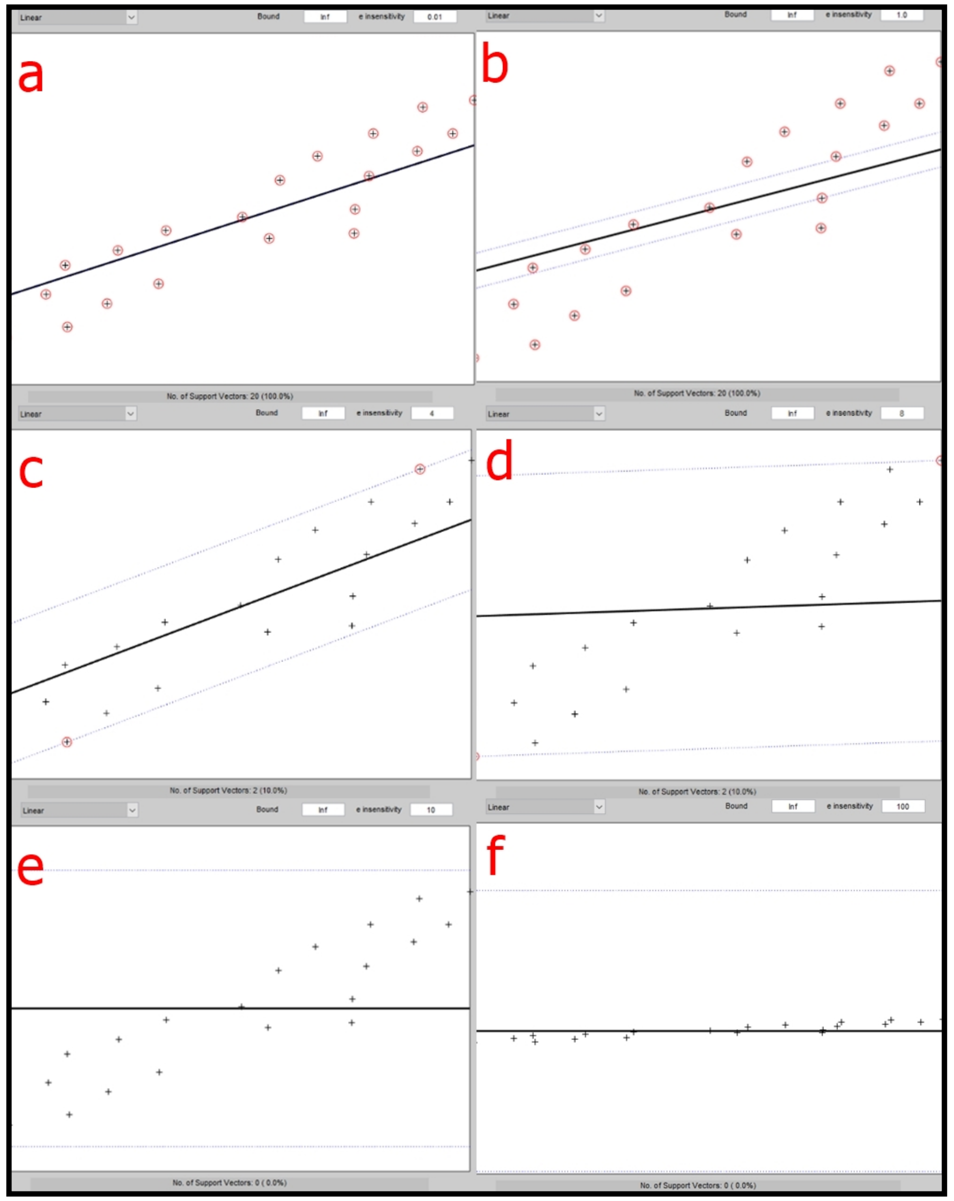


fig 2

1.1.3 Influence of bond

Performance is affected by the error penalty parameter for the wrongly classified datapoint.

a) small bound value —> small error penalty —> great freedom to increase |w| —> generalizes better
b) large bound value —-> big error penalty —> may overfits

Fig. 3:
Bound —> shape and orientation of decision line.
1) sensitivity = 4.0, —> bound = 4 (2 SVs) == BEST MODEL
2) decrease sensitivity —> narrower margin —> more overfitting —> more SVs.
3) increase bound —> loss of generalizatio
4) increase bound + decrease sensitivity —> largest number of SVs (here 20).

1.1.4 Sparsity property of solution in SVM

The SVM's solution is a set of support vectors and "sparseness of the solution" means that the number of support vectors increases more slowly than linearly when the problem size increases. Support vectors are the samples with non-zero coefficients. Keeping only the non-zero coefficients from all the rest is essentially the sparsity property of SVMs. Keeping a high number of support vectors decreases generalization of the SVM leading to overfitting and an increase in the computational power needed to determine the decision function.

Similarly, keeping a very low number of support vectors vastly increases the generality of the decision function. The e insensitivity parameter acts as our control in keeping the optimal number of support vectors. As it can be seen in Figure 2 and 3 above, a high value of e (4.0) decreases the number of support vectors to as low as 2. Therefore, to get an optimal regression function a moderately high value of e should be used (depending on the distribution of the dataset).

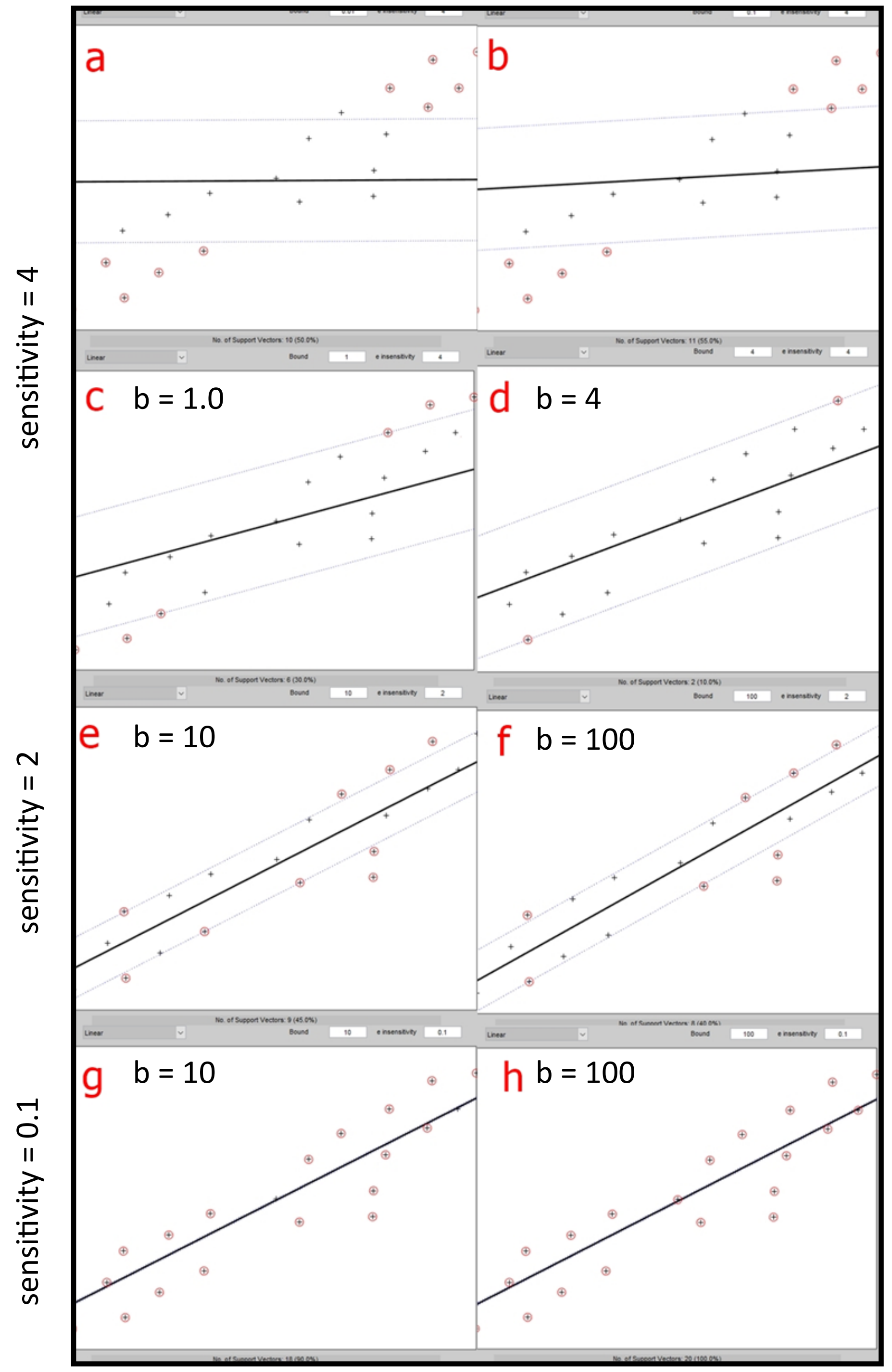


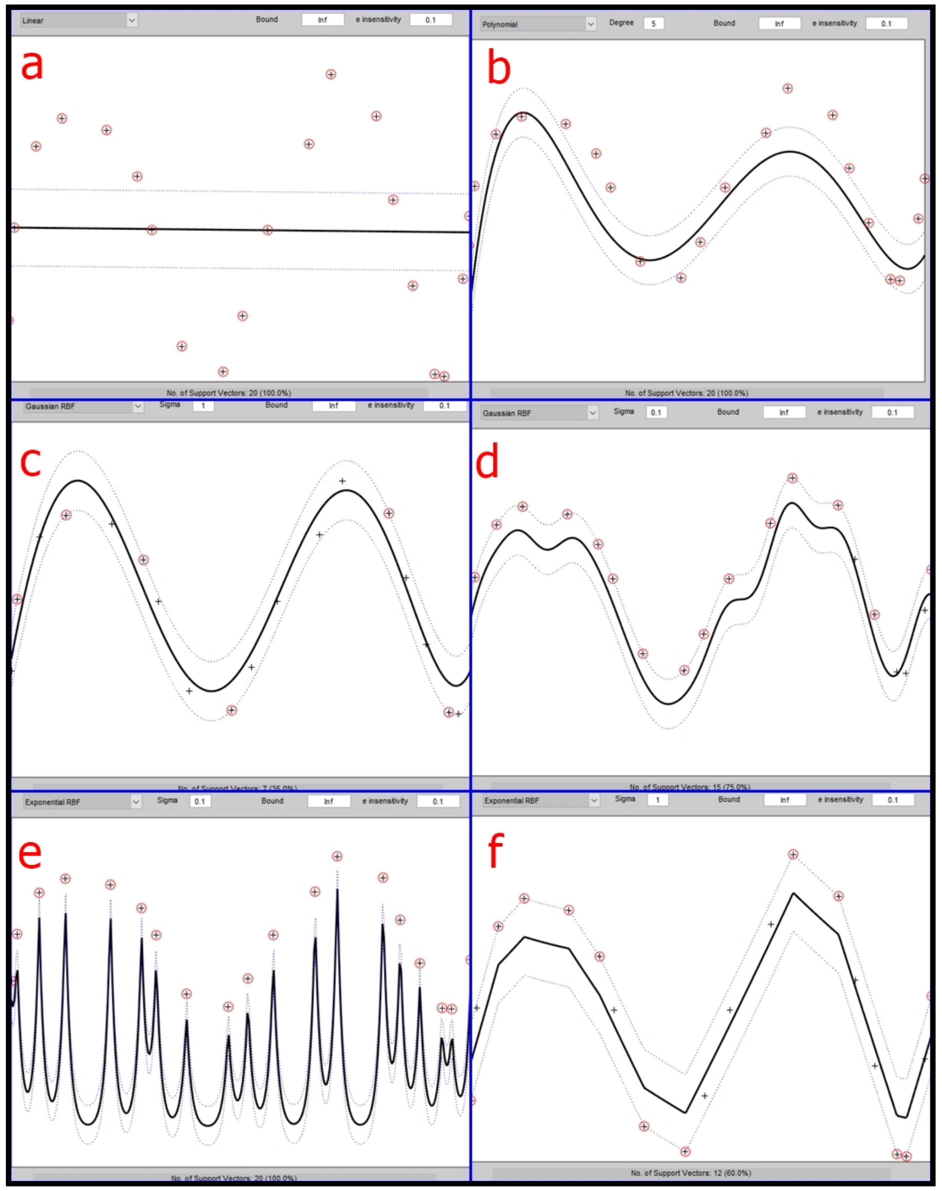
fig 3

1.1.5 Best kernel suited for more challenging dataset

Creating a more challenging dataset of 20 datapoints and running different kernels at sensitivity 0.1. Our data is not linearly separable and thus we can see from Figure 4a, that linear kernel in this case is not a good choice. Polynomial kernel of degree 5 is used, we see an overfitting on data with high number of support vectors i.e. 20. With sigma set at 1.0 the Gaussian RBF performs the best with a more generalized model and classifying data with 7 support vectors which goes up to 15 if the value of sigma is reduced to 0.1. Using the same hyperparameters values of 'e' and sigma, Exponential RBF performs less better, we see that the data is overfit, and we have a high number of support vectors. Therefore, for more challenging and smaller non-linear dataset, Gaussian RBF performs the best.

1.1.6 Difference between regression and classical least squares fit

The main difference is that the classical LS-SVM is used for classification problems where the number of classes is known and the testing datapoints are expected to belong to either of the known classes. The SVM regression on the other hand is used to determine functional relationship between the datapoints which is very useful in function estimation and time-series prediction problems. Another difference is that in LS-SVM the parameters are tuned to minimize the mean-squared error between the dataset and the decision boundary. The number of parameters is set by the complexity of the decision boundary and other than regularization, there is no real way to control this complexity. On the other hand, in support vector regression we can control the model complexity in terms of the number of support vectors by changing the 'e' parameter, choice of kernels and regularization.



a: Linear — Bound: Inf — e insensitivity: 0.1 — No. of Support Vectors: 20 (100.0%)

b: Polynomial — Degree: 5 — Bound: Inf — e insensitivity: 0.1 — No. of Support Vectors: 20 (100.0%)

c: Gaussian RBF — Sigma: 1 — Bound: Inf — e insensitivity: 0.1 — No. of Support Vectors: 7 (35.0%)

d: Gaussian RBF — Sigma: 0.1 — Bound: Inf — e insensitivity: 0.1 — No. of Support Vectors: 15 (75.0%)

e: Exponential RBF — Sigma: 0.1 — Bound: Inf — e insensitivity: 0.1 — No. of Support Vectors: 20 (100.0%)

f: Exponential RBF — Sigma: 1 — Bound: Inf — e insensitivity: 0.1 — No. of Support Vectors: 12 (60.0%)

fig 4

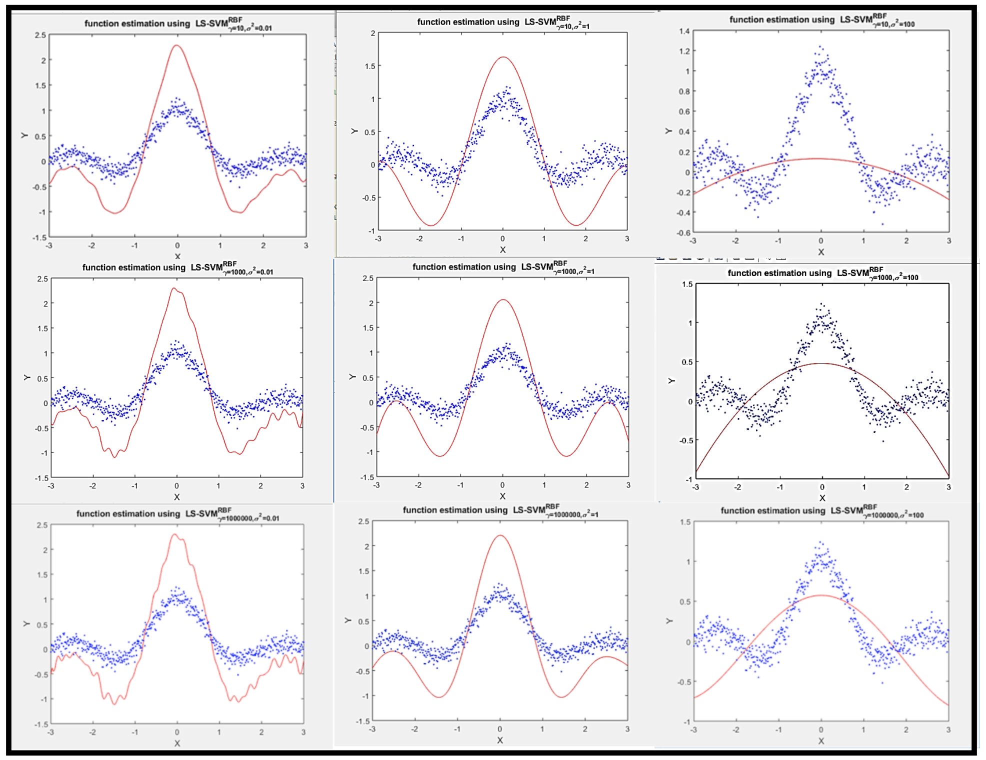
1.2 A Simple Example: Sinc Function

1.2.1 Function estimation and MSE for (gam, sig^2) combinations

Fig 5: results of SVM trained with different values of gam and sig^2 on a SINC function with little added noise. High and low value of gam was tested with high and low values of sig^2.

Fig 6: changing the Sig^2 parameter has a greater effect on the curve fitting on to the datapoints than the gam parameter. With Sig^2 we get smoother fit increasing it 0.1 to 1 but increasing beyond that gives a high error between actual and predicted values.

Using very low gam and sig^2 or very high values of these parameters does not give the best result. We get the best fit when Sig^2 is 1 and gam is 1000 (plot 5) i.e. the middle values from the given range of values.
However, there can be multiple combinations of these parameter values that give the most optimal results, so one can also use Grid Search along with Cross Validation to optimize the parameters for the best model.



different values of gam and sig2 **fig 5**

1.2.2 Hyperparameter tuning

Tuning the model with tunelssvm using RBF kernel, we get tuned [gam, sig^2] values of [12.404 0.320537] with simplex and [13.153 0.286948] with grid search.
The performance of these values is estimated by performing 10-fold cross-validation.

Fig6: there isn't much difference in results; simplex has a slightly lower error of 0.0103 compared to grid search with an error of 0.0105.
The tuned parameters' values are higher when using grid search especially for gam value.
The output function is clearly generalizing the dataset.
The simplex optimization performs faster than grid search.

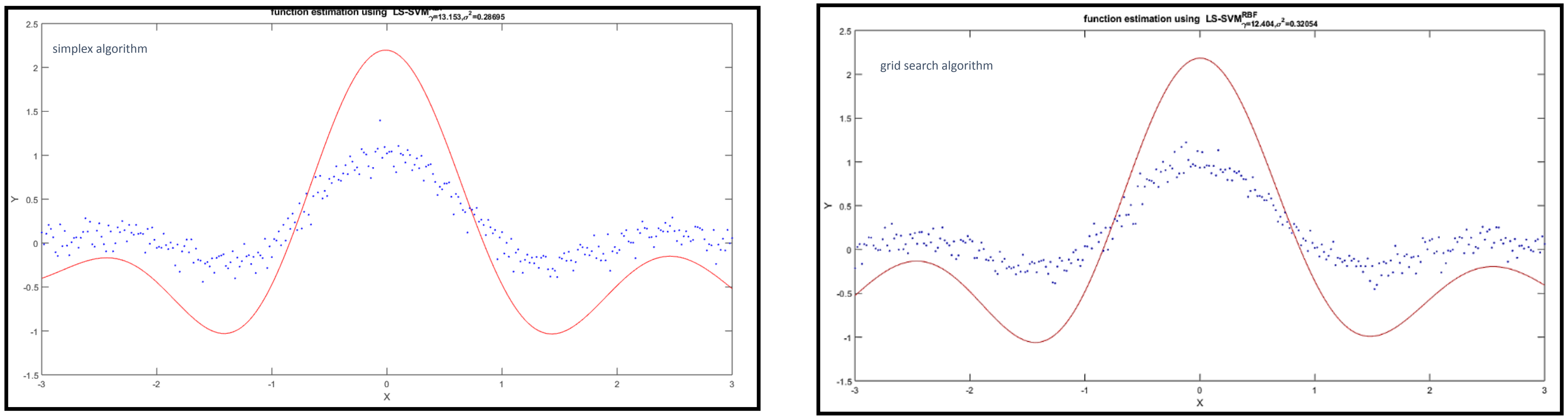


fig 6

1.3 Application of the Bayesian Framework

1.3.1 Illustrate this scheme by interpreting the function calls

Together with the observed data, prior distributions are converted to posterior distributions using Bayes' theorem. Bayesian model-based optimization: choose the next input values to evaluate based on the past results to concentrate the search on more promising values. In classical inference we are usually, only interested in inference about the parameters at the highest level to which the model is specified. In a Bayesian context the highest level is regarded as providing constraints or empirical priors that enable posterior inferences about the parameters in lower levels. The optimization by bayesian framework estimates the posterior probabilities of model hyperparameters on the different inference levels:

First level: In the first level one optimizes the support values alpha 's and the bias b. It computes the cost associated with the posterior of the model parameters. Function crit_L1 = bay_lssvm ({ Xtrain , Ytrain , 'f', gam , sig2 }, 1) describes the outputs on the first level.
Second level: In the second level one optimizes the regularization parameter gam. The cost associated with the posterior of the regularization parameter is computed. Function crit_L2 = bay_lssvm ({ Xtrain , Ytrain , 'f', gam , sig2 }, 2) describes the outputs on the second level. The level one parameter i.e. alpha and the bias become the prior in second level to predict the gamma.
Third level: In the third level one optimizes the kernel and its parameter sig2. The cost associated with the posterior of the chosen kernel and kernel parameters is computed. Function crit_L3 = bay_lssvm ({ Xtrain , Ytrain , 'f', gam , sig2 }, 3) describes the outputs on the first level. The gamma predicted in second level become the prior in level 3 to predict the Sig2.

1.3.2 Parameter tuning using the bayesian framework

In order to obtain LS-SVM model (with the RBF kernel), we need two extra tuning parameters: γ (gam) is the regularization parameter, determining the trade-off between the training error minimization and smoothness of the estimated function. σ2 (sig2) is the kernel function parameter. In Grid Search and Random Search, we try the configurations randomly and blindly. The next trial is independent to all the trials done before. In contrast, automatic hyperparameter tuning forms knowledge about the relation between the hyperparameter settings and model performance in order to make a smarter choice for the next parameter settings. One of the approaches for automated hyperparameter tuning is the Bayesian framework.

Re-sampling approaches, such as cross-validation to decide values of these hyperparameters, are very expensive when many parameters are involved. Typically, Bayesian methods are regarded as suitable tools to determine the values of these hyperparameters. The computation considers the estimated noise variance and the uncertainty of the model parameters, estimated by Bayesian inference.

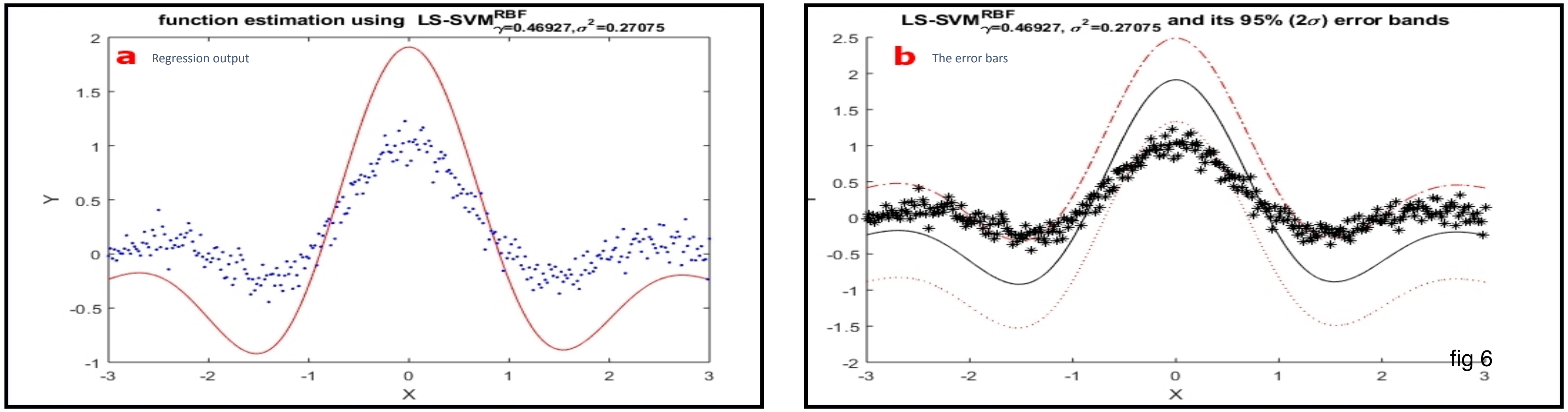


fig 7

1.3.3 Automatic Relevance Determination (ARD)

ARD determines the most relevant inputs for the LS-SVM within the Bayesian evidence framework. This is done by assigning a different weighting parameter to each dimension in the kernel and optimizing this using the third level of inference. In each step, the input with the largest optimal sig2 is removed (backward selection). For every step, the generalization performance is approximated by the cost associated with the third level of Bayesian inference. ARD is based on backward selection of the inputs based on the sig2s corresponding in each step with a minimal cost criterion. Working with the 'RBF kernel', the kernel parameter is rescaled appropriately after removing an input variable.
In our example problem, the three dimensions are ranked as the first dimension to be of highest ranking, followed by the third dimension and then the second dimension ranks the least. This way only the highest-ranking dimension is selected, which is the first one in our case. Using the cross-validation with assigned weights we can replicate this procedure. This can be done by eliminating the dimensions with the highest cost, based on the results of the optimized hyperparameter values.
Next, we used the LS-SVM command to calculate error bars based on a sample training data set.

Fig 8: The first is the plot of the actual data points (blue dots) compared to the function estimated to predict those points (red line), .
In the second plot the black line covered by black '+' signs is the predicted function and data, while the red dotted lines represent the 95% error bar; basically, the area in which 95% of the data resides.

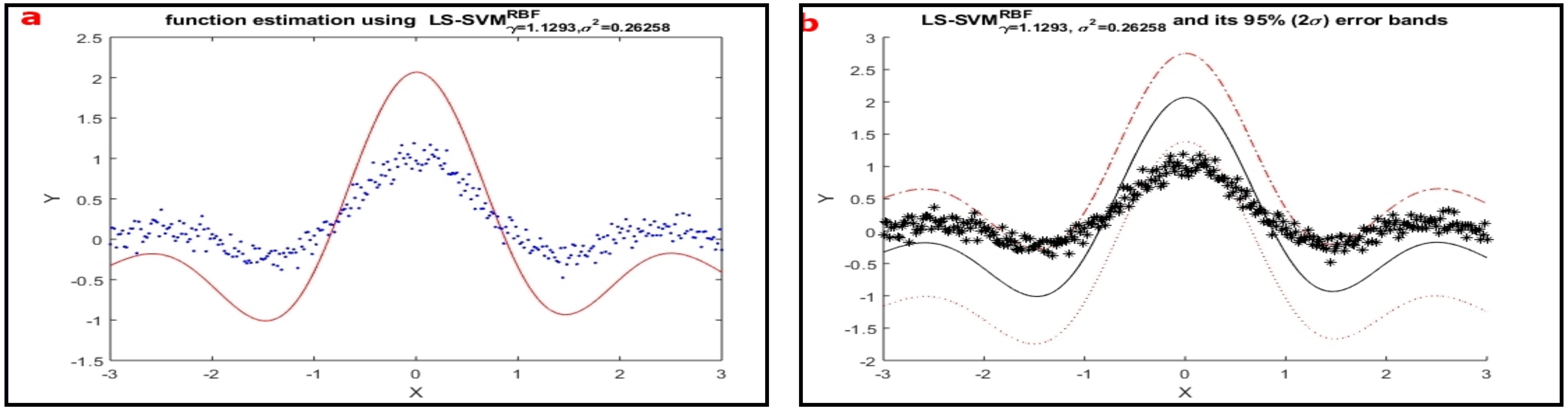


fig 8

1.4 Robust Regression

1.4.1.1 Non robust vs Robust version

A dataset was made to see the influence of noise and outliers on the regression model. The regression model to try to fit the outliers onto the data resulting in loss of generalization and a misfit in the regression outputs. The added noise has a significant impact because each datapoint is then like a new class, and thus every datapoint has an equal weightage. This causes the regression to try to lean towards the outliers. Robust regression helps the SVM regression to tolerate the presence of noise and outliers.

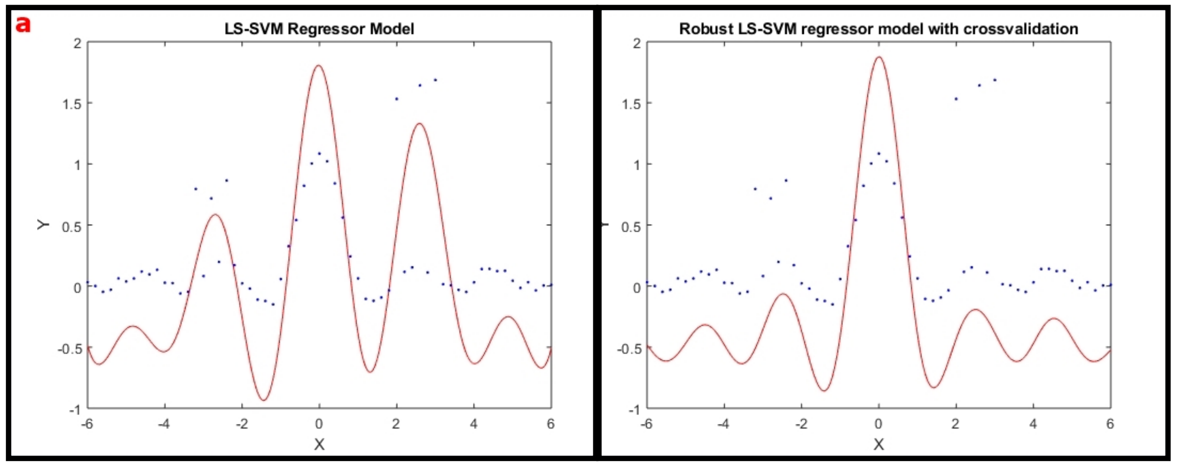


fig 9

Fig.9: results of robust regression on the same example dataset.
These results clearly show the effectiveness of robust regression in identifying the outliers and ignoring them for a better regression output.

1.4.1.2 MSE vs MAE

In MAE we calculate the residual for every data point, taking only the absolute value of each so that negative and positive residuals do not cancel out. We then take the average of all these residuals. MAE describes the typical magnitude of the residuals. A small MAE suggests high model performance at prediction, while a large MAE suggests the model has certain problems.
While each residual in MAE contributes proportionally to the total error, the error grows quadratically in MSE. This ultimately means that outliers in our data will contribute to much higher total error in the MSE than they would the MAE. Therefore, our model will be penalized more for making predictions that differ greatly from the corresponding actual value. This is to say that large differences between actual and predicted are punished more in MSE than in MAE.
MSE is better in estimating model's prediction errors when we want the model to take outliers into account more. Conversely, in this example we wanted that outlier residuals won't contribute as much to the total error as MSE, and so MAE is preferred. Mean Absolute Error (MAE) in our example minimizes the errors that are less than 1 and maximizes the errors that are more than 1. The outliers would have a greater error and their MAE would be significantly larger than the original points, making it easy to identify them.

1.4.2 Results with different weight functions

Changing the weighting function generally does not have much of an impact on the final output of the robust regression.

FIg 10: For sig^2 and gam values, weighting function 'wmyraid' [13259.9012 0.112895316] has the highest values while 'wampel has the lowest values [23.364 0.0399477]. We know that high values of gamma produce highly flexed decision boundaries, and low values of gamma often results in a decision boundary that is more linear. While smaller sigma tends to make a local classifier, larger sigma tends to make a much more general classifier. Therefore, we see slightly better results with wymiard weighting function.

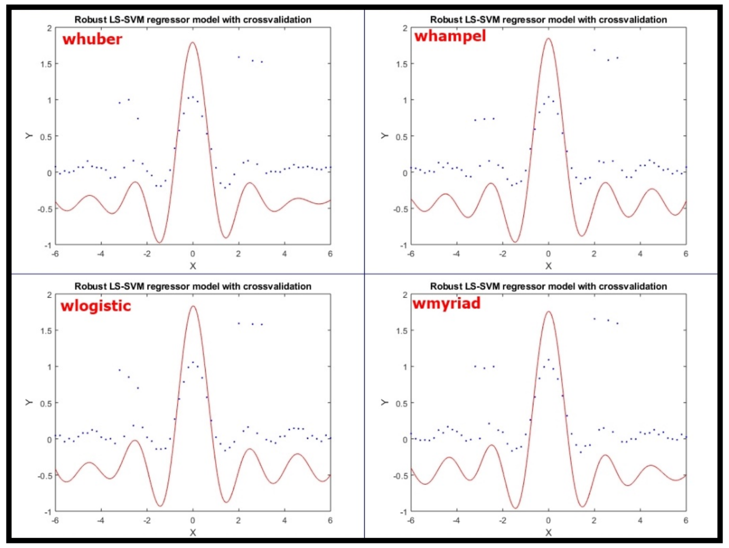


fig 10

2 Homework

2.1 Intro to Time Series Prediction

In this section we use SVMs to predict the future values of time-series dataset. The dataset used is logmap, a chronologically ordered sequence of 150 training and 50 each test data points with and without noise.

Fig.11 : the training and test data plots.
In time series prediction, unlike regression, the sequence of values generated by the function is assumed to be dependent on the time of generation and on preceding values. The input values are assumed to be time units and the output values are assumed to be in chronological order.

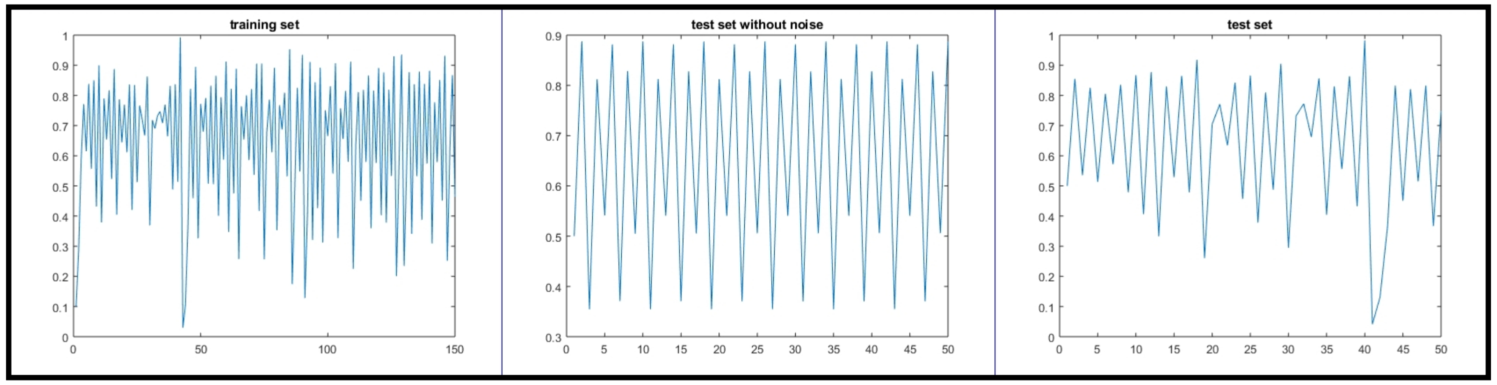


fig 11

2.1.1 Application: Santa Fe Laser Dataset

The Santa Fe Laser Dataset is a time-series set dataset containing continuous sets of 1000 and 200 samples each for training and testing. We try to find a good model for predicting the future values of this series with varying values of order and other hyperparameters to get an optimal tuning parameter setting for the model.

Fig 12: The training set of the time-series.

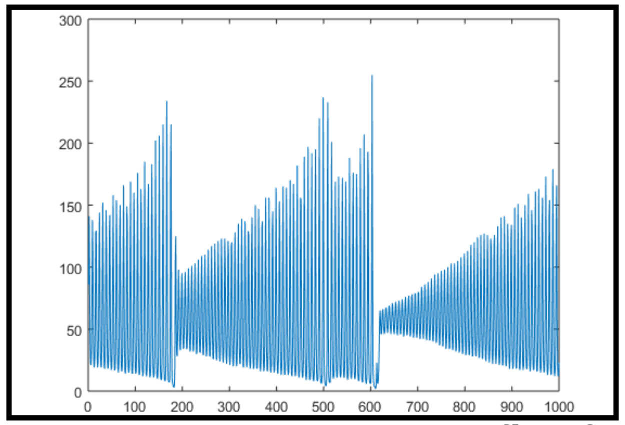


fig 12

2.1.3 Order 50 for the auto-regressive model

A larger value of 'order' increases the model complexity and makes the prediction computationally expensive and slower. Keeping these considerations in mind, we use an order value of 50 for predicting the time-series. We will test these assumptions keeping gam and sig^2 constant at 10 and just the value of order is optimized by having a fixed validation set and checking the trained model's mean squared error on the validation set.
Figure 13: error rates with varying values of order.

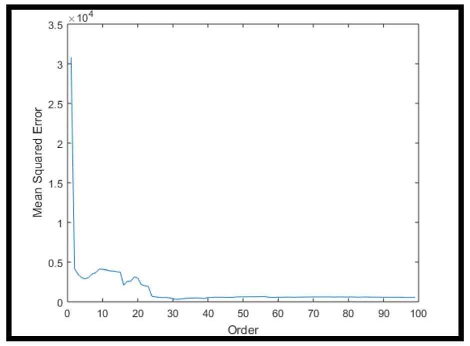


fig 13

The graph shows a trend of decreasing mean squared error with increasing values of order. There is a sharp decrease in the error, followed by a moderate gradient and at the end a steady horizontal line. The lowest scores for the mean squared error of 312.17 is obtained at order 32. The mean squared error at order 50 is 624.11, which is not very high considering the overall error rates of this time-series but still these are higher than the error score of when order was 32.
Fig14: results of using these orders on the model training.

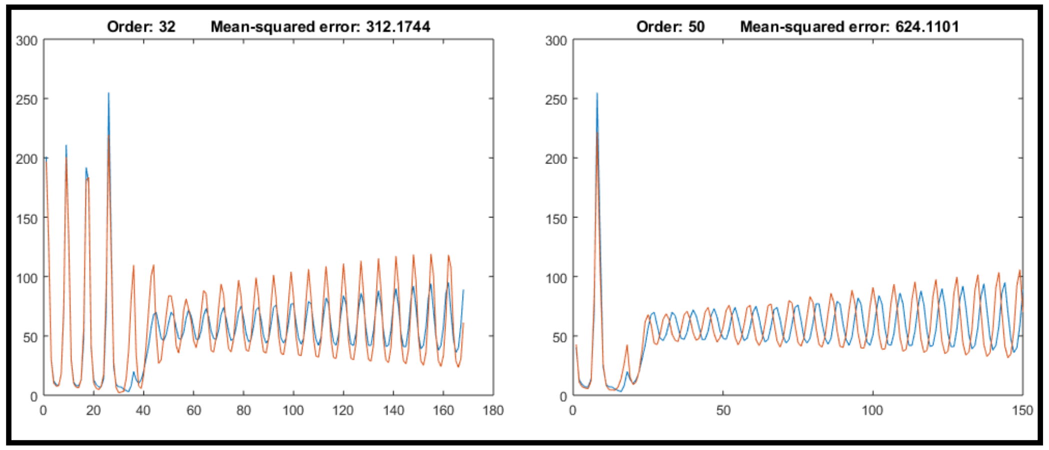


fig 14

We now use 32 and 50 as o preferred values for the order parameter. Order 32 has the lowest mean squared error and to avoid increasing complexity of model while 50 is taken as the end point to keep the model complexity within a certain limit.
With this tuned order parameter range we now optimize the gam and sig^2 parameters by using the tunelssvm function and a fixed validation set. First both these parameters are tuned on order 32 and then on order 50. Results are shown in figure 15.

2.1.3 Using the performance of this recurrent prediction on the validation set

Figure 15 shows that the mean squared error increases for both order 32 and 50, compared to our previous results. We now adopt a new strategy for optimizing the gam and sig^2 parameters. Mean square error scores are used as the metric for evaluating the scores of combinations of values for gam and sig^2 on a fixed validation set. Optimal values of gam=200 and sig^2=2.2 are computed with mean squared error of 569.2 for the order 50, and for the order 32 we get gam and sig^2 values of 110 and 1.6 with a mean squared error of 460.35.

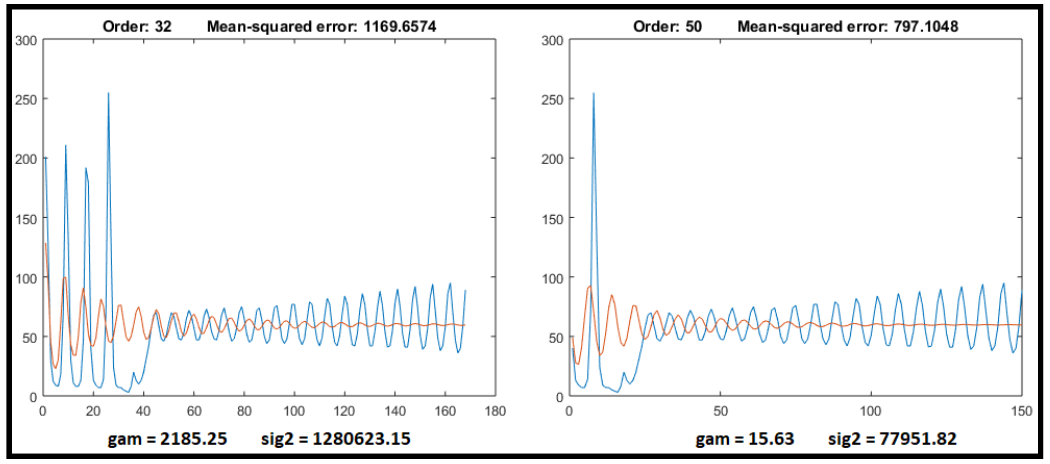


fig 15

2.1.4 Tuning parameters order, gam and sig2

By using multi-fold cross-validation scores for the parameter tuning, gam and sig^2 values of 12867.96 and 0.052 are obtained for the order 32, with a mean squared error of 483.61. Using order 50 we get gam 2915.7 and sig^2 5.165 with a mean squared error of 491.24. These results show that the tunelssvm function is not very effective in optimizing the parameters in our example, as we get higher error rates. This can be due to the large values of sig^2 it gives which can be because it does not find better initial estimates. The optimized parameters obtained from the fixed validation sets and the cross-validation sets turn out to be more accurate, with the cross validated scores being slightly better.

In conclusion, the with order 50 relatively low error scores are achieved without overfitting or unnecessarily complicating the model. However, careful optimization could lead to better values for this parameter (like the order=32 we got earlier). The perfect combination of all the parameters is hard to find but with a good knowledge of the significance of and function of each parameter, we can get more accurate initial estimations for improved optimization. Finally, optimizing these parameters on a fixed validation set offers consistency and uniformity of the testing conditions.