KU LEUVEN

MAI

Support Vector Machines

Exercises

r0685182
Yi Zhao

08.2018

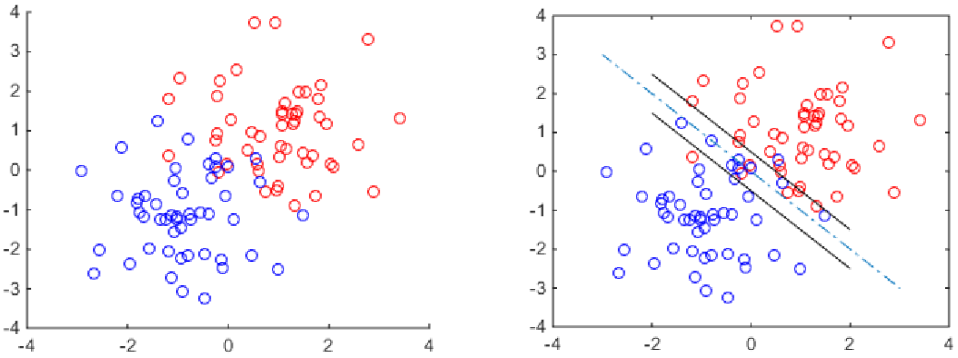
Support Vector Machines: Methods and Applications

Exercise Session I

1 Exercise Session 1: Classification

1.1 A Simple Example: Two Gaussians

A data-set is constructed from two classes of Gaussians with the same covariance matrices in Figure 1-(a):



(a) (b)

Figure 1: Two Gaussians and the classifier

Given the two gaussians, to estimate the optimal classifier using lines, we can use non-separable case linear SVM classifiers by adding slack variables when minimizing the margin. As for the question under which conditions this construct is optimal/valid, it might be achieved by solving the primal problem:

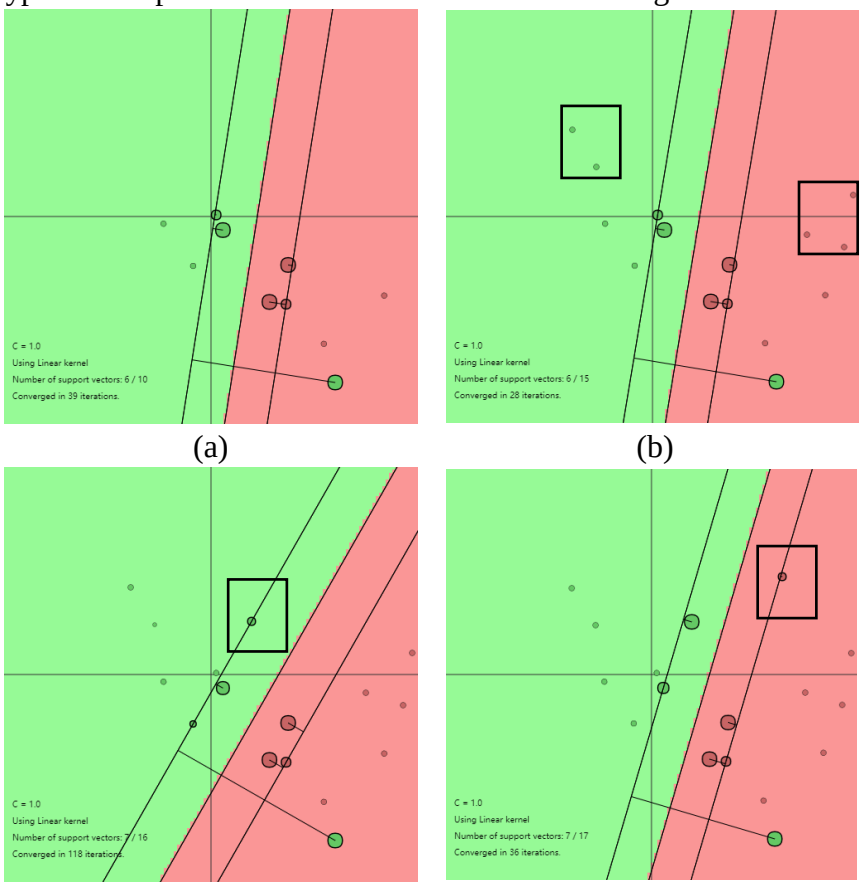
$$\min_{w,b,\xi} J_P(w,\xi) = \frac{1}{2}w^T w + c\sum_{k=1}^{N}\xi_k$$
$$s.t.\ y_k(w^T x_k + b) \geq 1 - \xi_k,\ \ \xi_k \geq 0, k = 1,\dots,N$$

where c is a positive real constant. The classifier may look like the dashed line in Figure 1-(b).

1.2 The Support Vector Machine

1. Use the linear kernel. Adjust the existing datasets to have at least 10 data points for each class. Figure 2 shows the changing process when adding data points to the classes. We can see:
   - It won't change much if we add data points of the same class to the positions far away outside the margin, comparing Figure 2-(a) and Figure 2-(b).
   - The slope of the linear classifier might change a lot if we add data points of the same class to the positions inside the margin, comparing Figure 2-(b) and Figure 2-(c).
   - The support vectors might be different (the new data points will work as support vectors), or their weights might change if we add data points of the same class to the positions inside the margin, comparing Figure 2-(c) and Figure 2-(d).

As for the question 'how drastically can classification boundaries change', it depends on where we add new data points and what class the new data point is. Adding it inside the margin results in more drastic change than adding it outside. And adding the opposite type of data points also results in more drastic change.

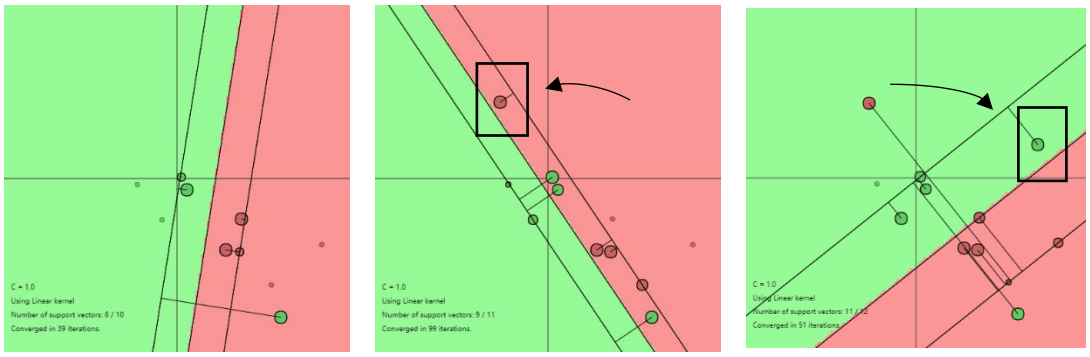


(a) (b)

(c) (d)

Figure 2: Adding data points to the class

2. If we add an outlying datapoint which lies on the wrong side of the classification boundary, it might work as a support vector and therefore change the direction of the classification hyperplane towards itself as well as the margin width, as shown in Figure 3:

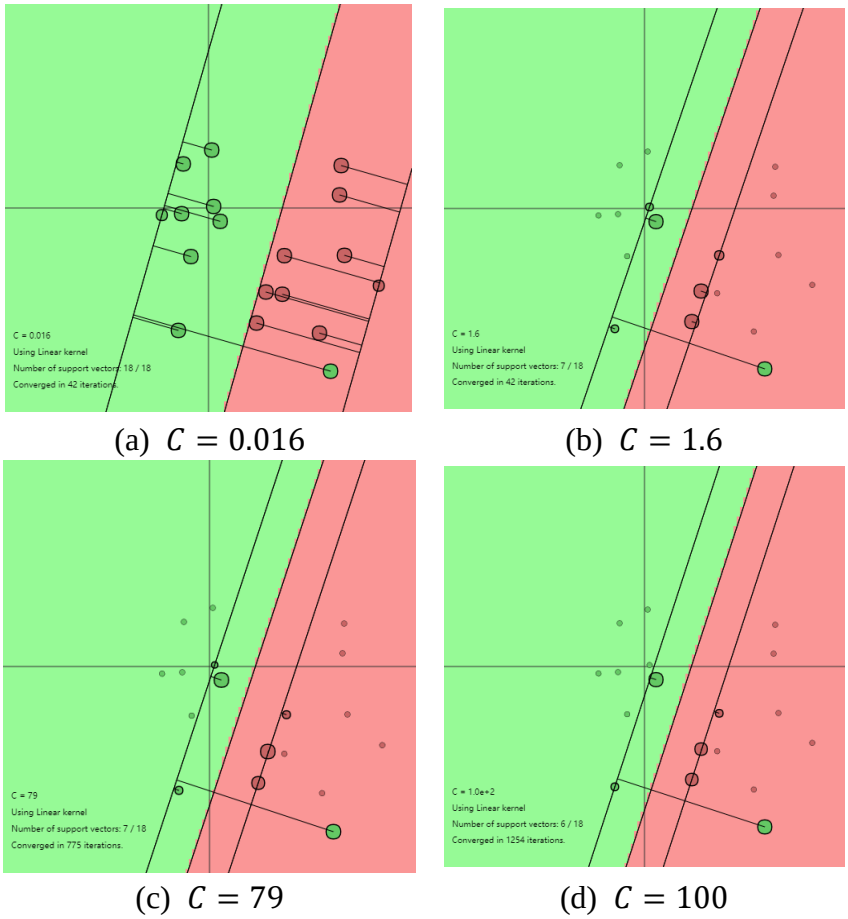


(a) (b) (c)

Figure 3: Adding an outlying datapoint on the opposite side

3. Let's discuss the influence of the regularization parameter C. Figure 4 displays 4 cases when C equals to 0.016, 1.6, 79 and 100. We can see when the value of C

increases from 0 to 100, the margin first narrows down and then doesn't change much. At the same time, the number of support vectors first decreases and then doesn't change much.



(a) $C = 0.016$ (b) $C = 1.6$

(c) $C = 79$ (d) $C = 100$

Figure 4: Classifiers with different values of regularization parameter *C*.

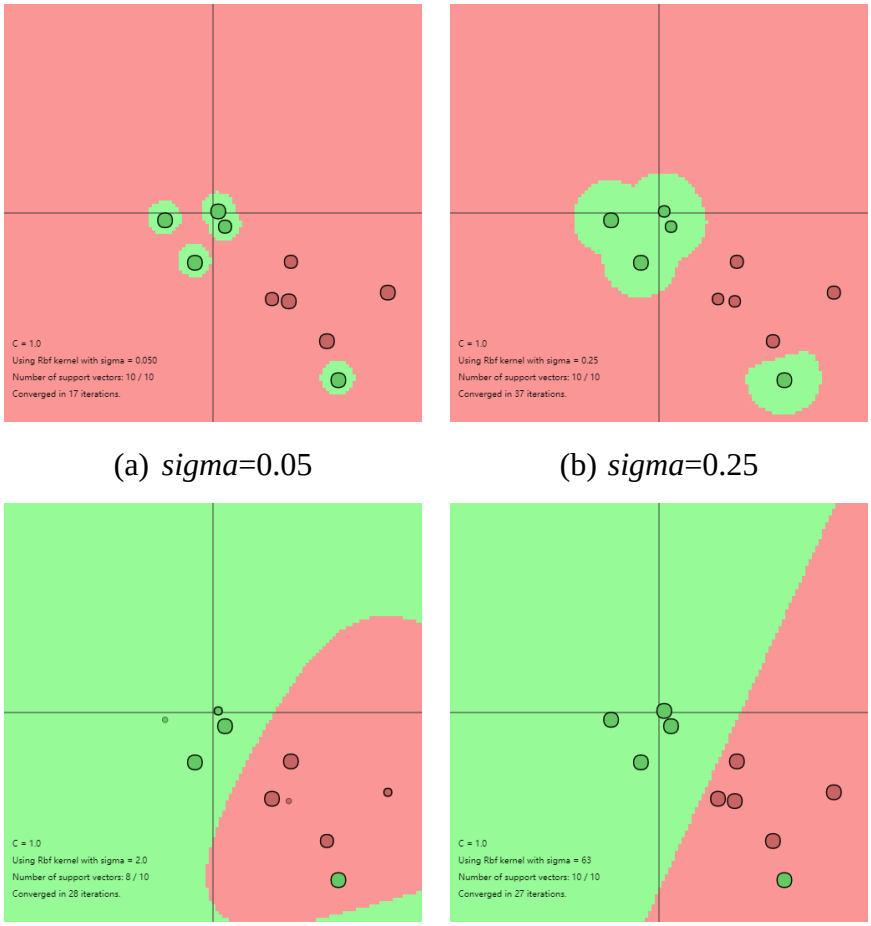
According to the formulations $\min_{w,b,\xi} J_P(w,\xi) = \frac{1}{2}w^T w + c\sum_{k=1}^{N}\xi_k, \quad s.t.\ y_k(w^T x_k + b) \geq 1 - \xi_k, \xi_k \geq 0, c > 0,$ C is a penalty term affecting the classification outcome in this way:

- A smaller value of C means it tolerates more misclassifications and allows more slack variables ξ_k, therefore, leading to a larger minimum margin.
- Similarly, a larger value of C means it tolerates less misclassifications and requires that more training data points being classified correctly. Therefore, it might lead to overfitting.

The role C plays is that it controls the trade-off between minimizing the norm of the weights and achieving a low error on the training data. It is a tuning parameter and we can determine its value by validation methods or applying VC bounds.

4. Switch to RBF kernel and compare the classification outcome to that of the linear case.
5. Change the RBF kernel $sigma$ hyperparameter. Figure 5 displays 4 cases when $sigma$ equals to 0.05, 0.25, 2.0 and 63. The intuition is:
   - when $sigma$ is large, the decision boundary tends to be linear;
   - when $sigma$ is small, the decision boundary becomes highly nonlinear, strict and sharp. It might lead to overfitting.

- besides, when $sigma$ increases, the number of support vectors decreases and then increases; the number of iteration times increases and then decreases.



(a) *sigma*=0.05 (b) *sigma*=0.25

(c) *sigma*=2.0 (d) *sigma*=63

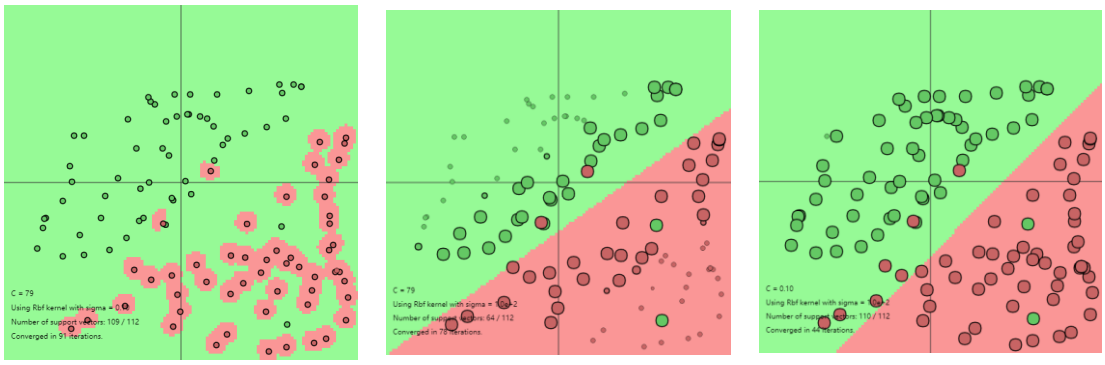
Figure 5: Classifiers with different values of *sigma*

The parameter $sigma$ affects the classification boundaries in this way: the smaller the $sigma$ is, the more nonlinear the boundary is. According to the equation $K(x,z) = \exp(-(\|x-z\|_2^2/\sigma^2)$, $sigma$ determines the width of Gaussian distribution. When the distance between x and z is much larger than $sigma$, the kernel function tends to be zero. Thus, if $sigma$ is very small (for example, Figure 5-(a)), when we estimate a new point of green class, only if the point within small distance to the green support vectors, it would be classified as green. Therefore, small $sigma$ models don't have good generalization.

Now we try to change both C and $sigma$ hyperparameters.

If the data is almost linearly separable, we can choose a more linear boundary and tolerate less misclassifications. Based on our former analysis, we'd better to choose a large $sigma$ value (for a linear boundary) and a large C value (for tolerating less misclassifications).

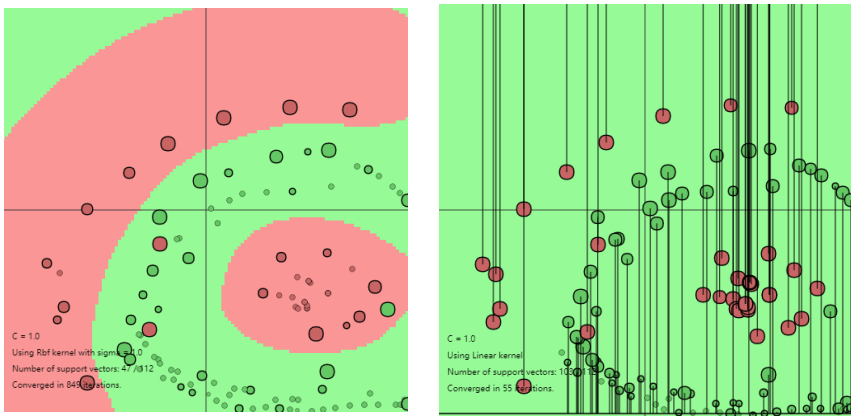
Figure 6 compares the cases of different C and $sigma$ combinations. We can see the classifier in Figure 6-(b) is the best one while the classifier in Figure 6-(a) is overfitting and the classifier in Figure 6-(c) is underfitting.



(a) $C = 79,\ sigma$=0.1 (b) $C = 79,\ sigma$=100 (c) $C = 0.1,\ sigma$=100

Figure 6: Different *C* and *sigma* combinations and corresponding classifiers

6. When the dataset is linearly non-separable with an overlapping region between classes, let's discuss the chosen kernel, C and $sigma$.
   1) We'd better to choose RBF kernel. As shown in Figure 7, RBF kernel works well while linear kernel fails to separate the two classes.



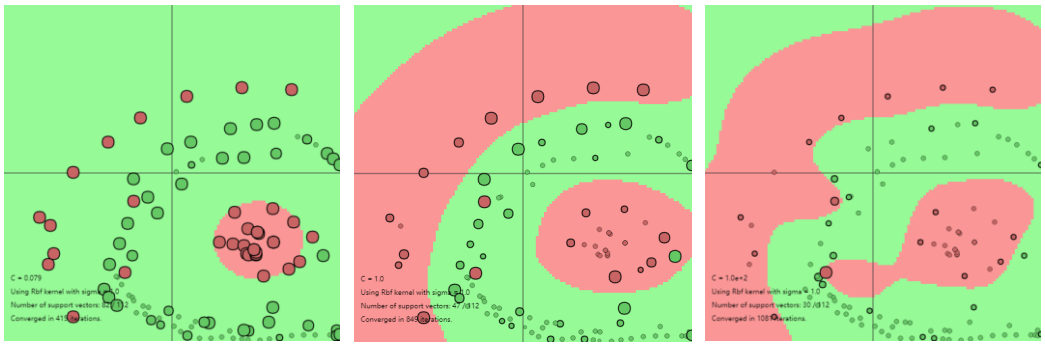
(a) RBF kernel (b) linear kernel

Figure 7: RBF kernel and linear kernel when data is linearly non-separable

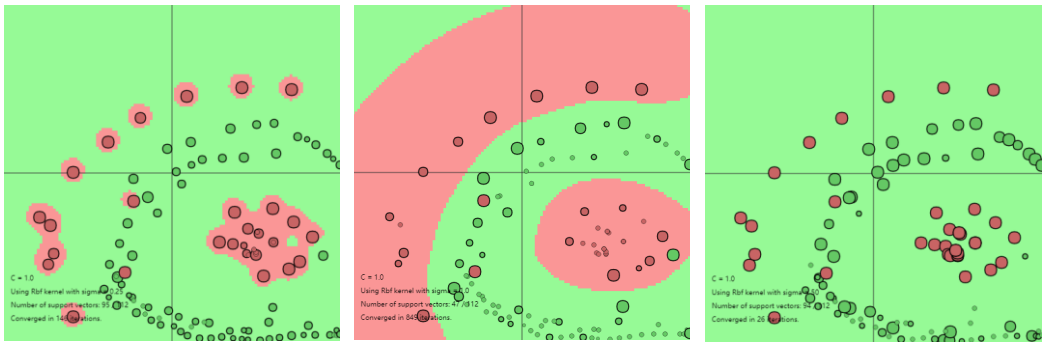
Tune the hyperparameters C and $sigma$ simultaneously. For this constructed dataset (linearly non-separable with an overlapping region), when both C and $sigma$ are not too small nor too large, the classifier has the best performance. As shown in Figure 8-(b). when $C = 1.0,\ sigma$=1.0, the classifier works well.

If C is too small, the classifier would tolerate too many misclassifications, which, leads to underfitting, as shown in Figure 8-(a). On the contrary, if C is too large, it requires a very small training error, which leads to overfitting, as shown in Figure 8-(c).

If $sigma$ is too small, it leads to overfitting, as shown in Figure 8-(d); if $sigma$ is too large, it leads to underfitting, as shown in Figure 8-(f).



$C = 0.079,\ sigma$=1.0 (a) $C = 1.0,\ sigma$=1.0 (b) $C = 100,\ sigma$=1.0 (c)



$C = 1.0,\ sigma$=0.25 (d) $C = 1.0,\ sigma$=1.0 (e) $C = 1.0,\ sigma$=50 (f)

Figure 8: Different C and *sigma* when data is linearly non-separable with an overlapping region

7. The role of support vectors is: support vectors is a subset of training samples, they fully specify the decision function. When we use neural networks, every training data point would be used for determining w and b. However, in SVM, only data points close to the decision boundary would be used for determining w and b. Suppose the original training data is linearly non-separable, we try to find the largest margin in the hyperspace. Datapoints whose mapping in the hyperspace maximizes the margin become support vectors. Usually, they are close to the decision boundary.
8. The corresponding importance (measured by the size of the bold-lined circle) of a support vector is the degree how much it 'supports' to find the decision boundary. It changes when C or $sigma$ changes. When C increases, it decreases, and when $sigma$ increases, it increases.

1.3 Using LS-SVMlab

1. For the $Iris$ dataset, the classifier is shown in Figure 9.
   Using the test set X_t and Y_t and comparing the simulation result Y_{test} with Y_t, the test error rate is 0.55.

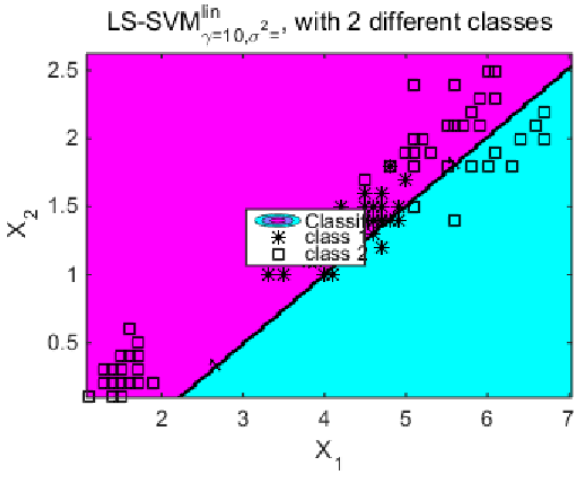
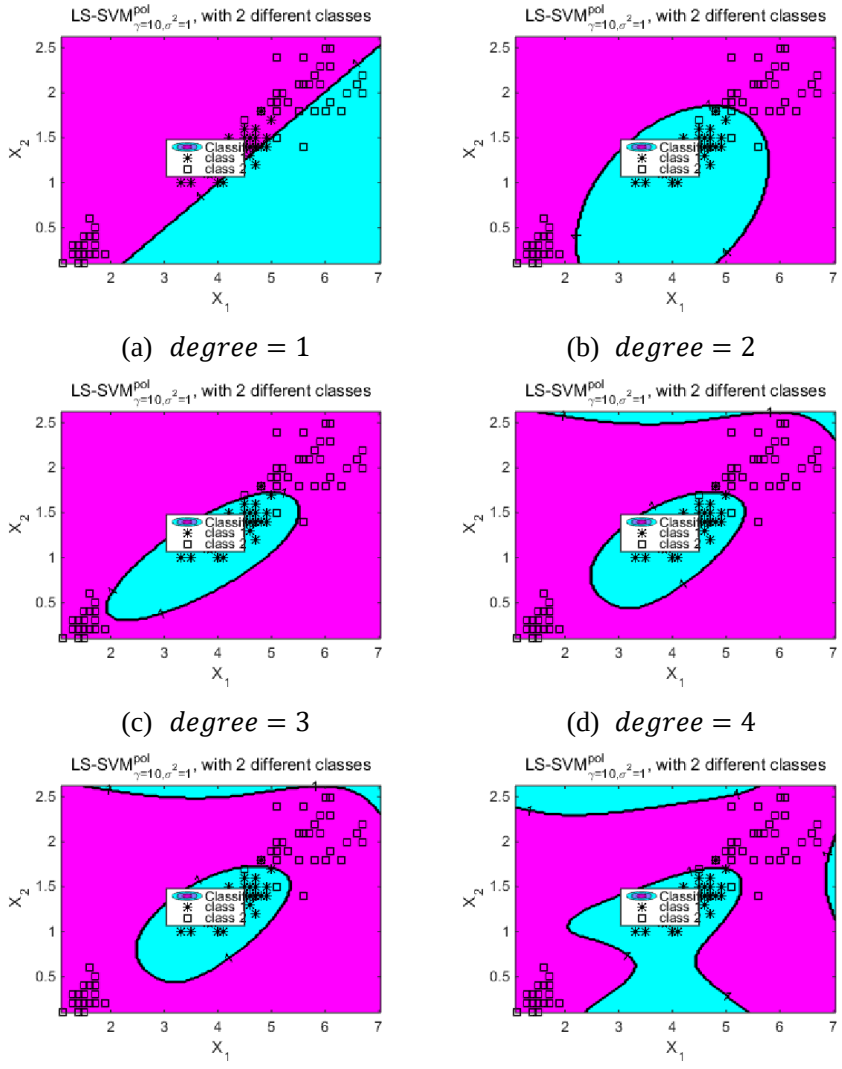


Figure 9: Iris dataset with linear kernel

Now we try the polynomial kernel with $degree = 1,2,3,4,5,6$ and $t = 1$ .The classifiers are shown in Figure 10.



(a) $degree = 1$ (b) $degree = 2$

(c) $degree = 3$ (d) $degree = 4$

(e) $degree = 5$ (f) $degree = 6$

Figure 10: Polynomial kernel with *degree*=1,2,3,4,5,6 and *t*=1.

For polynomial kernel, when $degree$ equals to 1, it is linear kernel. When $degree$ increases, the decision boundary becomes more non-linear. Low *degree* leads to underfitting and high *degree* leads to overfitting. It corresponds to the changes in $sigma$ of RBF kernel. Increasing $degree$ is similar with decreasing $sigma$, and vice versa.

2. Now we focus on the RBF kernel with parameter the bandwidth σ^2. Try different $sig2$ values in the range [1:0.5:100] and calculate the corresponding test error (the ratio of misclassified test data points), as shown in Figure 11.

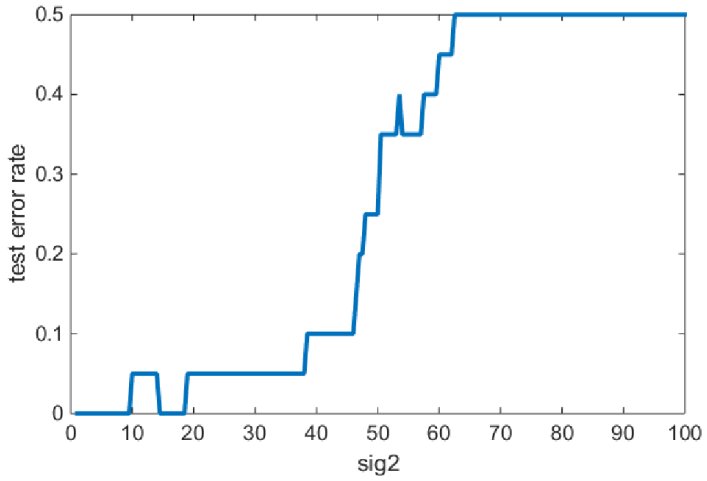


Figure 11: Test error rate when *sig2* changes (*gam*=10)

3. Now let's look at the regularization constant gam. Based on question 2, I would fix $sig2$ to be 16. Choose different gam values in the range [1:0.5:100] and calculate the corresponding test error

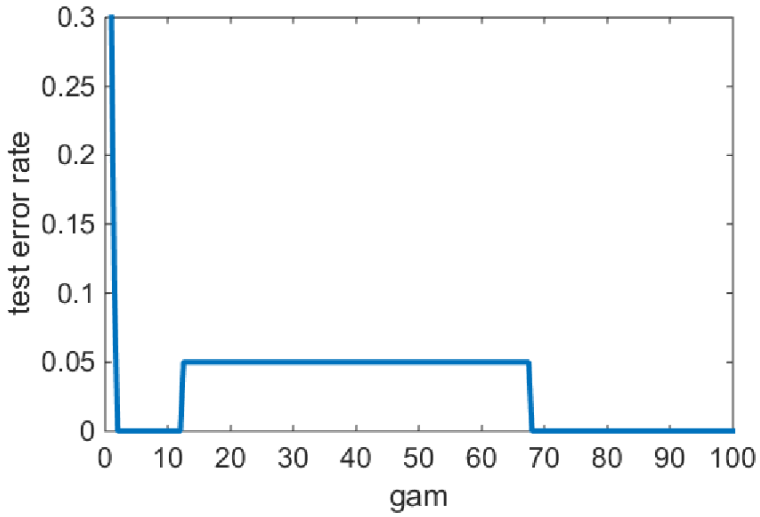


Figure 12: Test error rate when *gam* changes (*sig2*=16)

According to Figure 12, when $sig2$ equals to 16, a *good* range for gam is [2, 12].

4. **1.3.1 The choice of the Hyper-parameters**

1. Validation set used to optimize a number of (tuning-) parameters cannot used again to measure the final model. It is because: if we use all points in dataset for training, we can't check whether this new learned model overfits or not.

Train a LS-SVM model with $Xtrain$ and $Ytrain$ and evaluate it on $Xval$.The plot of the performance (validation error rate) with respect to several values of gam and $sig2$ ($gam = 0.1, 1, 10, 100$ and $sig2 = 0.1, 1, 10, 100$) is shown in Figure 13.

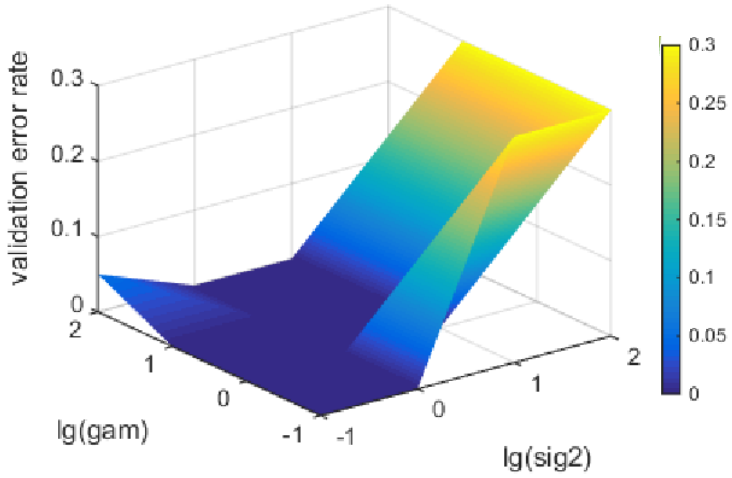
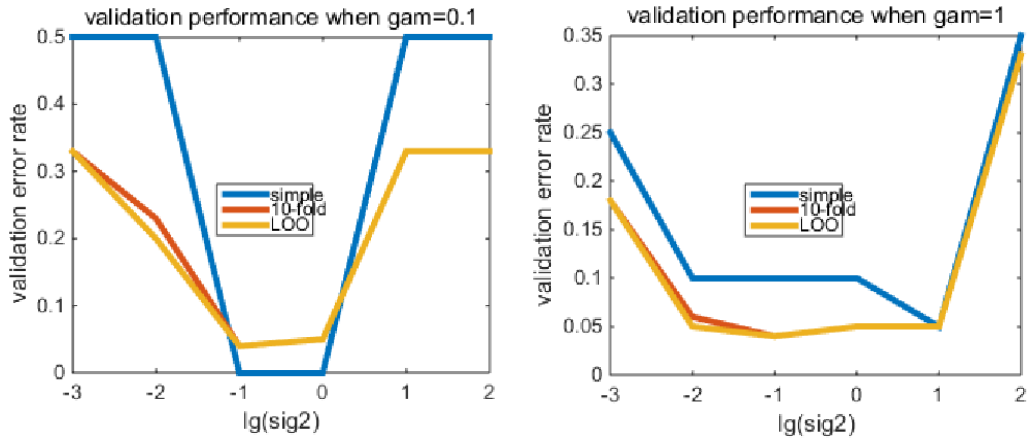


Figure 13: Validation error rate for different *sig2* and *gam* values

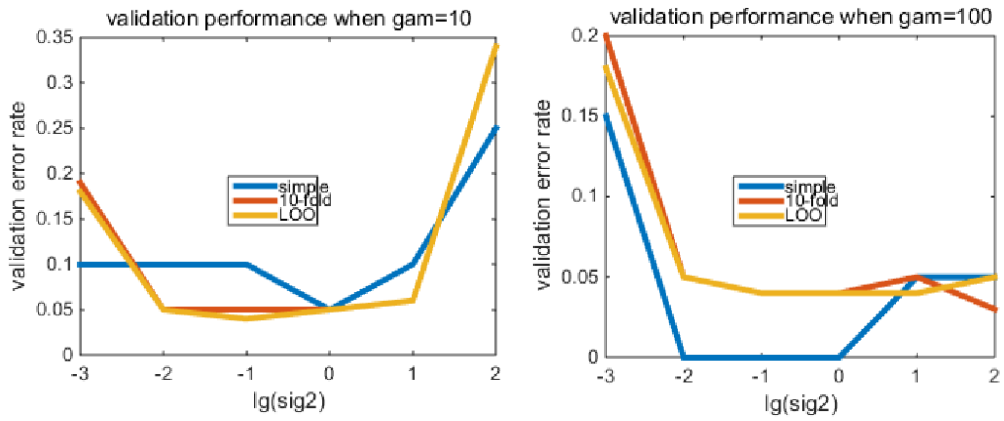
From Figure 13, we may conclude that for this dataset, when gam and $sig2$ both are around 1, the validation error rate is smallest. The result is reasonable. For $sig2$, if it is too small, it would lead to overfitting; and if it is too large, it would lead to underfitting. For gam, if $sig2$ is small, larger gam leads to higher validation error rate (more overfitting); if $sig2$ is large, smaller gam leads to higher validation error rate (more underfitting).

2. Now let's perform crossvalidation using 10 folds. To represent this technique, I would change the $Xtrain, Ytrain, Xval$, and $Yval$ for 10 times (the omitted sections for validation without overlap) and perform the same operations as in question 1.

We prefer cross validation over simple validation. It is mainly because cross validation makes use of all the data samples. Also change $crossvalidation$ procedure for $leaveoneout$. Figure 14 shows the performance of these three validation methods.



(a) (b)

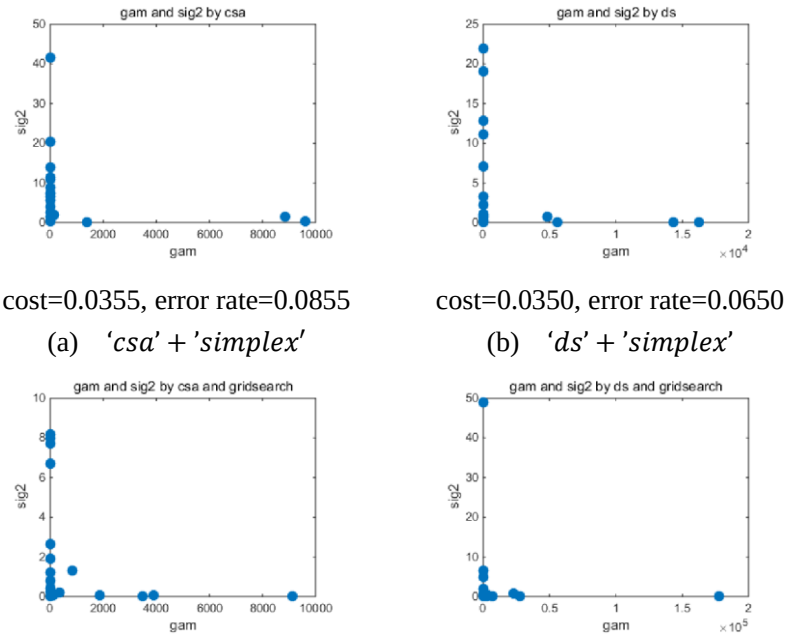


(c) (d)

Figure 14: Validation methods comparation

For this dataset, 10-fold validation and LOO have similar performance. And both of them are better than simple validation. When the parameters keep same, the simulation result of simple validation varies a lot, which indicates that it might not stable. Leave one out cross validation is better when we have a small set of training data. In this case, we can't really make 10 folds to make predictions on using the rest of data to train the model. If we have a large amount of training data on the other hand, 10-fold cross validation would be better, because there will be too many iterations for leave one out cross-validation.

3. Use $tunelssvm$ procedure to optimize the hyperparameters. Change different parameters like 'csa' vs. 'ds' and '$simplex$' vs. '$gridsearch$'. For each combination, perform the procedure for 20 times and we can get 20 (gam, $sig2$) pairs respectively, as shown in Figure 15. For each case, the average cost and validation error rate (using 10-fold validation as in question 2) were calculated.



cost=0.0355, error rate=0.0855

(a) 'csa' + '$simplex$'

cost=0.0350, error rate=0.0650

(b) 'ds' + '$simplex$'

cost=0.0335, error rate=0.0650

(c) 'csa' + '$gridsearch$'

cost=0.0335, error rate=0.0675

(d) 'ds' + '$gridsearch$'

Figure 15: *gam* and *sig2* distributions with different tunelssvm methods

We may observe:

- For the four cases, their cost and validation error rate are similar.
- 'ds' is more likely to produce outliers than 'csa' (Figure 15-(c) and (d)).
- '$gridsearch$' produces ($gam, sig2$) pairs with smaller variance than '$simplex$' (Figure 15-(b) and (d)).

The obtained hyperparameters may differ a lot. It is because the start points differ a lot and the converged final points are based on former points. In addition, 'ds' is more likely to drop into local minimum points than 'csa'.

4. One alternative way to judge a classifier is by using the ROC curve. Using the training set is not recommended. It is because when we measure whether a model has good generalization or not, we need to use test dataset. And when we decide the parameters of the model, we need to use validation dataset.

1.4 Homework Problems

1.4.1 The Ripley Data-Set

1. The plotted data is shown in Figure 16. Ripley dataset consists of two classes where the data for each class have been generated by a mixture of two Gaussian distributions

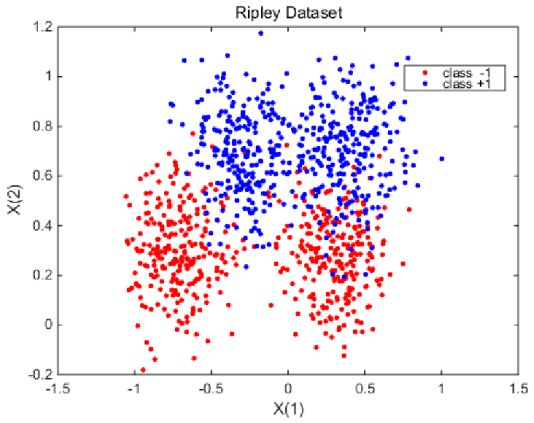
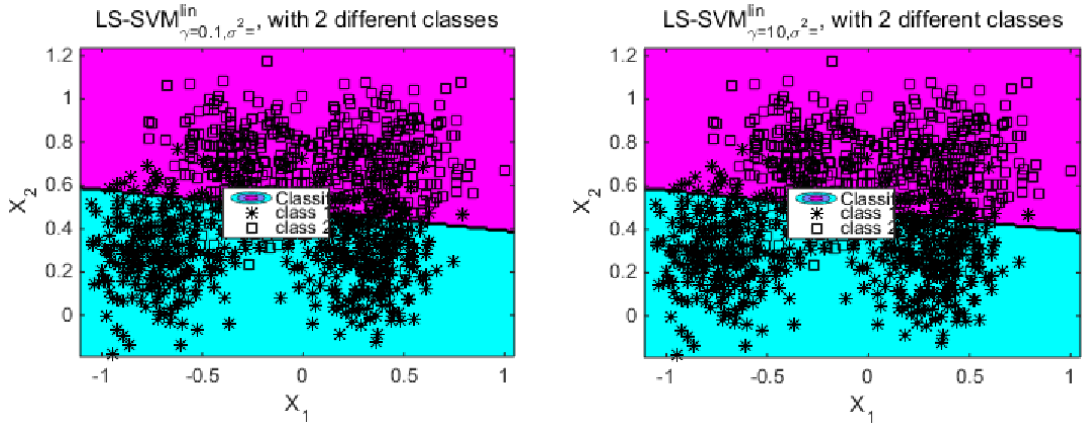


Figure 16: Ripley dataset

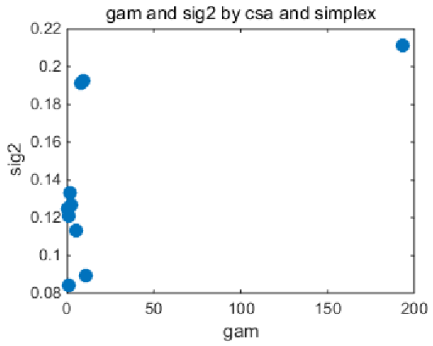
2. Try a linear model. Set the value of gam to be 0.1, and 10. As shown in Figure 17, it is not linear-separable.



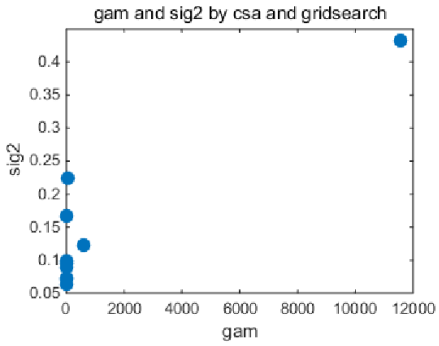
(a) *gam*=0.1 (b) *gam*=10

Figure 17: Linear classifiers for Ripley dataset

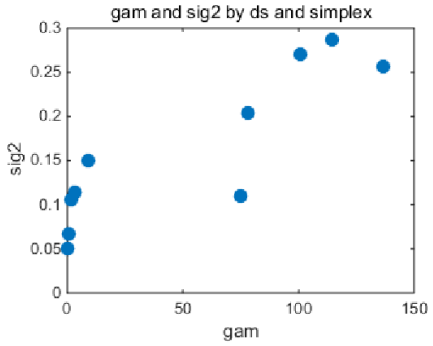
3. Try the RBF kernel and tune its parameter. Run 10 times for each 'csa' vs. 'ds' and '$simplex$' and '$gridsearch$' situation. Figure 18 plotted the $(gam, sig2)$ pairs. The variance of cost is also calculated.



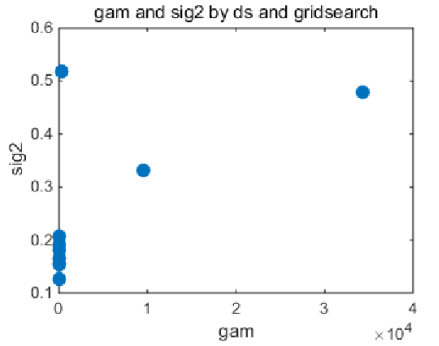
(a) 'csa' + '$simplex$'
$Var(cost)$=2.0556e-06



(b) 'csa' + '$gridsearch$'
$Var(cost)$=1.7333e-06



(c) 'csa' + '$gridsearch$'
$Var(cost)$=6.6667e-07

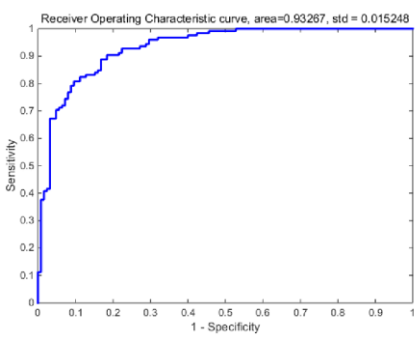


(d) 'csa' + '$gridsearch$'
$Var(cost)$=1.2889e-06

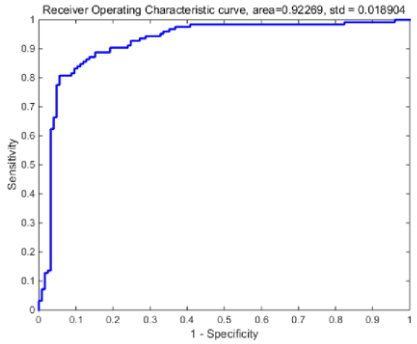
Figure 18: (gam, sig2) pairs using different tuning methods

The tuning is acceptable although the time complexity is a little high. From Figure 18, we conclude that gam changes a lot and $sig2$ doesn't change much. After calculating the variance of cost, we can see the performance (cost) almost doesn't change.

4. Consider the ROC curve. Figure 19 shows the ROC curves using linear kernel and RBF kernel separately. The value of gam and $sig2$ is gathered according to the $tunelssvm$ ('csa' + '$simplex$') procedure.



(a) linear kernel



(b) RBF kernel

Figure 19 ROC curves using linear kernel and RBF kernel

From Figure 19, we can see the areas under the curve are similar, but RBF-kernel case has a better sensitivity. Therefore, I would choose RBF kernel.

5. As for the final model. It uses RBF kernel with $gam = 909.7582,$, and $sig2 = 0.7511$ by $tunelssvm$ ($'csa' + 'gridsearch'$). The classifier is shown in Figure 20.

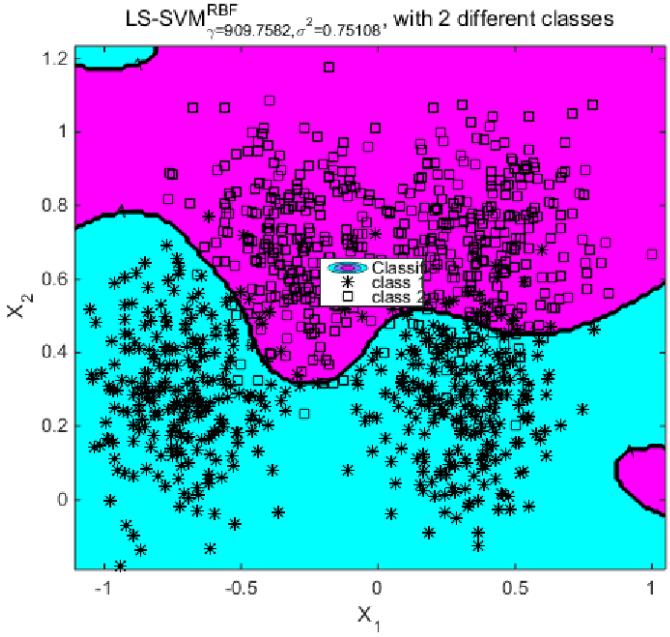


Figure 20: The final classifier for Ripley dataset

It is a good classifier with test error rate 0.1320.

Support Vector Machines: Methods and Applications

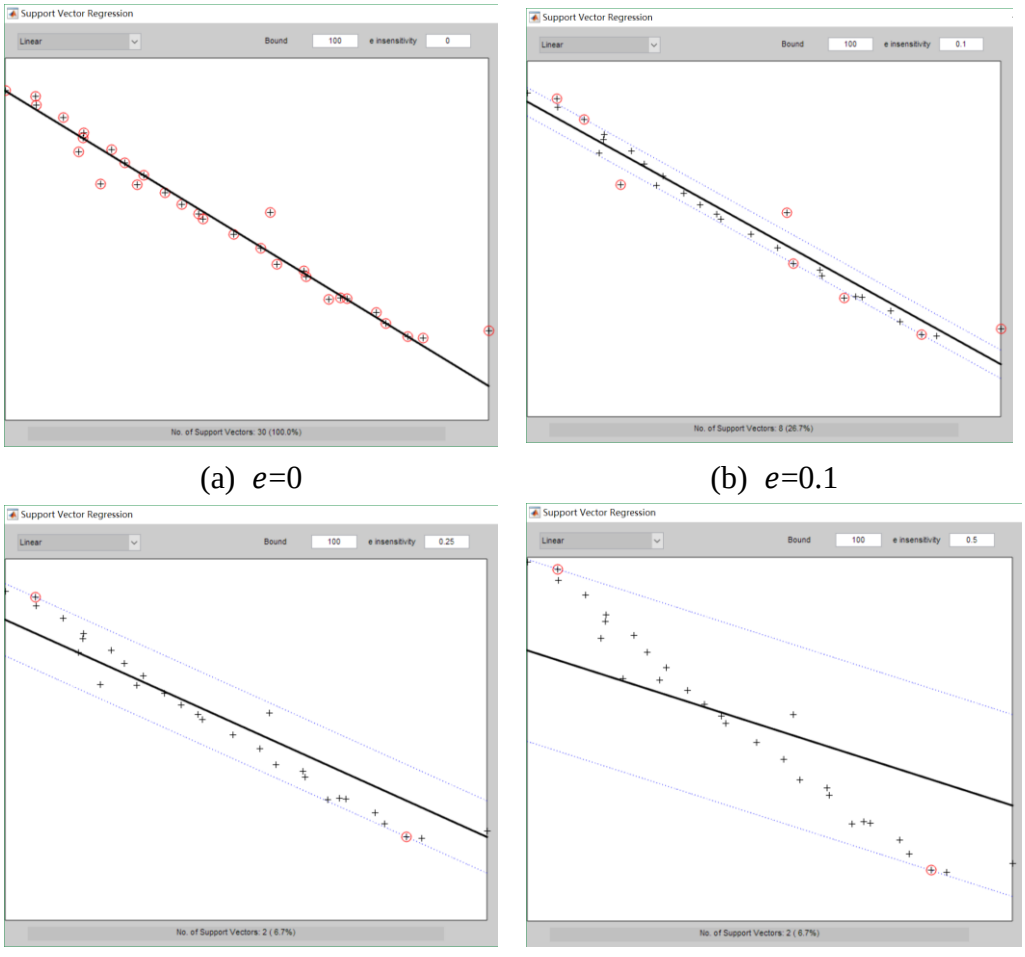
Exercise Session II

2 Exercise Session 2: Function Estimation and Time-series Prediction

2.1 The Support Vector Machines for Regression

A dataset where a linear kernel is better than any other kernel is constructed for SVM regression. There are two main parameters: e (the tube accuracy ϵ) and $bound$ (regularization term, determines the amount up to which deviations from the desired ϵ accuracy is tolerated). We now analyze their influence.

Figure 1 displays four cases with different values of e.



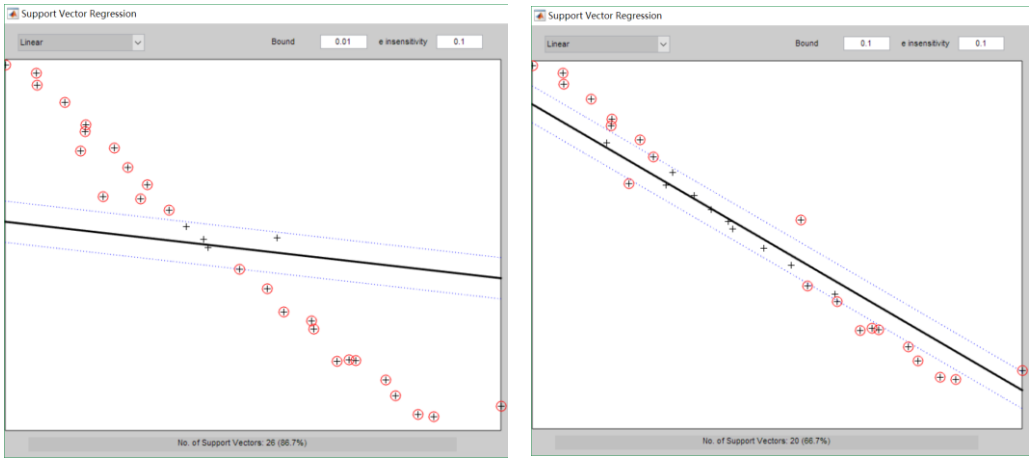
(a) e=0 (b) e=0.1

(c) e=0.25 (d) e=0.5

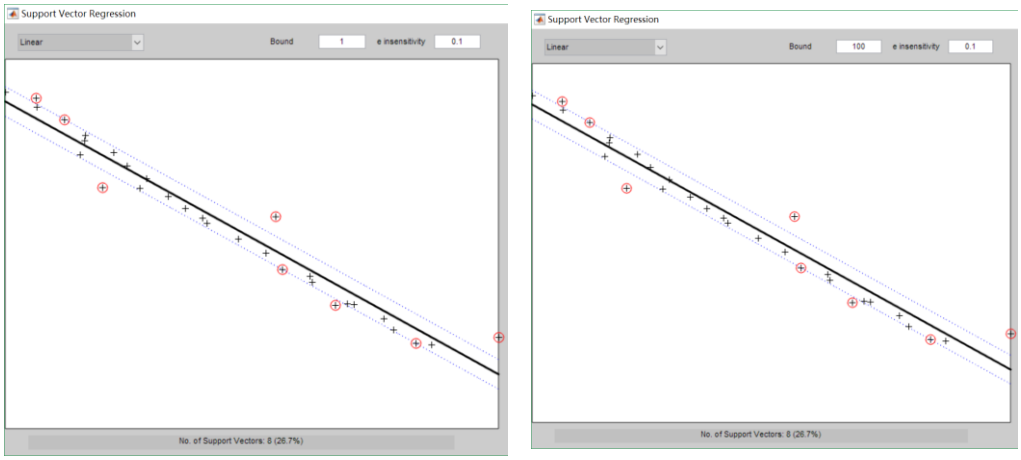
Figure 1: Regression with different values of e ($bound$=100)

We can see: when e increases, the tube becomes wider and the fraction of support vectors decreases.

Similarly, Figure 2 displays four cases with different values of $bound$.



(a) $bound$=0.01 (b) $bound$=0.1



(c) $bound$=1 (d) $bound$=100

Figure 2: Regression with different values of *bound* (*e*=0.1)

We can see: when $bound$ increases, the fraction of support vectors decreases, too. In addition, the value of $\|w_0\|^2$ increases.

SVM regression has sparsity property. For those data points inside the ϵ −tube, the slack variables ξ_k, ξ_k^* equal to 0 and $|y_k - w^T\varphi(x_k) - b|_\epsilon = 0$. In the Dual Problem, it is required that

$$\alpha_k(\epsilon + \xi_k - y_k + w^T\varphi(x_k) + b) = 0$$
$$\alpha_k^*(\epsilon + \xi_k^* + y_k - w^T\varphi(x_k) - b) = 0$$

Therefore, $\alpha_k = 0$ and $\alpha_k^* = 0$, which explain the sparsity.

The loss function of SVM regression is different from that of a classical Least Square's fit, as shown in Figure 3.

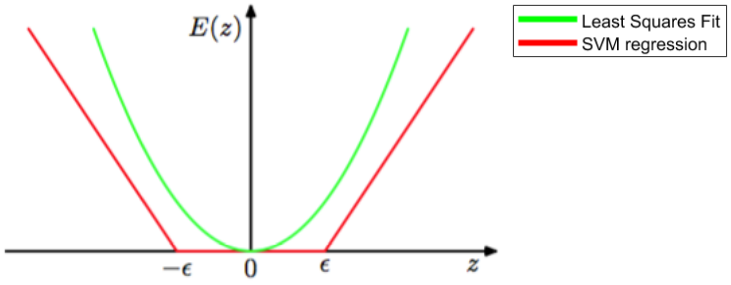


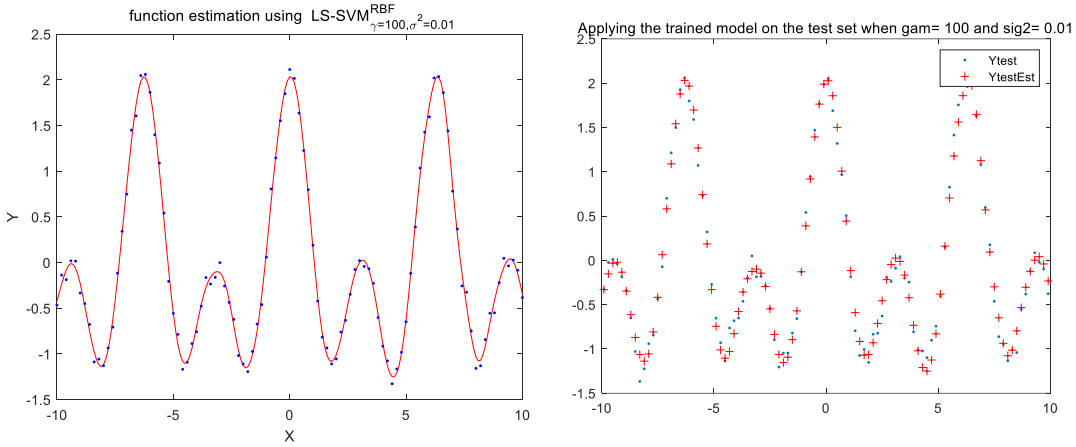
Figure 3: Different loss functions

2.2 A Simple Example: Sum of Cosines

In this exercise section, LS-SVM is used. The function to be estimated is:

$$y = \cos(x) + \cos(2x)$$

With gam in the range of [0.01, 0.1, 1, 10, 100] and $sig2$ in the range of [0.01, 0.05, 0.1, 0.5, 1], the results on the training set and test set can be visualized. An Optimal performance occurred when gam=100 and $sig2 = 0.01$, as shown in Figure 4.

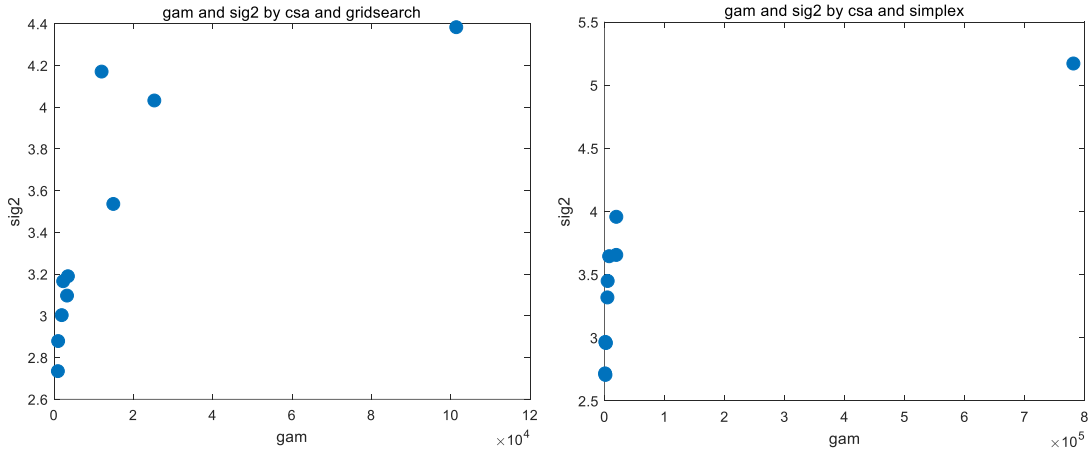


(a) Function estimation on training set (b) Apply the model on test set

Figure 4: Results on the training and test set when an optimal performance occurs

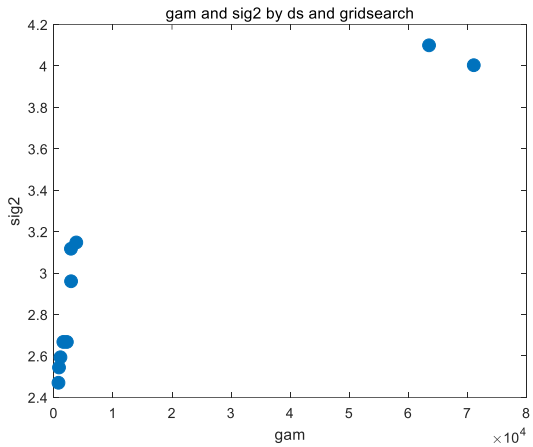
2.3 Hyper-parameter Tuning

The optimization of the hyper-parameters is done with the function $tunelssvm$ on the train dataset. With different combinations of $optfun$ ($'gridsearch'$ vs. $'simplex'$) and $global10ptFun$ ($'csa' vs. 'ds'$), run the $tunelssvm$ command for 10 times at each case.



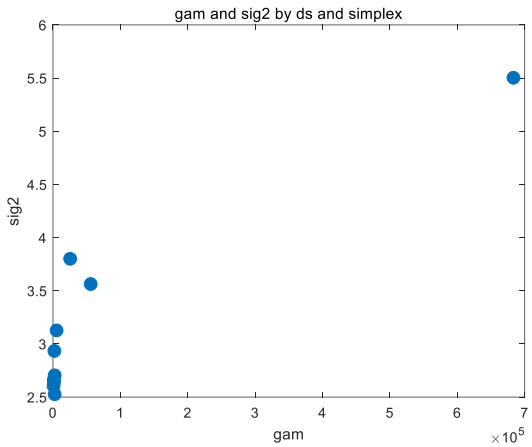
avg_cost=0.0129 , var_cost=4.10E-07
avg_time=0.9599, var_time=0.0259
(a) 'csa' + '$gridsearch$'

avg_cost=0.0133 , var_cost=5.60E-07
avg_time=0.5370, var_time=0.0032
(b) 'csa' + '$simplex$'



avg_cost=0.0108 , var_cost=3.27E-07
avg_time=0.8141, var_time=0.0029

(c) 'ds' + '$gridsearch$'



avg_cost=0.0112 , var_cost=1.33E-07
avg_time=0.5174, var_time=0.0029

(d) 'ds' + '$simplex$'

Figure 5: *tunelssvm* with different combinations of *optfun* and *global10ptFun*

According to Figure 5, we may conclude:

- The four cases have similar $cost$.
- '$gridsearch$' produces more condensed hyperparameter pairs than '$simplex$' while '$simplex$' costs much less time (around 40% reduction).
- The hyperparameter pairs produced by $'csa'$ and 'ds' are similar, but 'ds' costs a little less time than $'csa'$ (around 10% reduction).

Therefore, 'ds' + '$simplex$' is the most fast method.

2.4 Application of the Bayesian Framework

2.4.1 Regression

The Bayesian framework now can be used to tune and to analyze the LS-SVM regressor. There are 3 levels of Bayesian inference for LS-SVM. In the first level, we optimize the support values $alpha$'s and the bias b. In the second level, we optimize the regularization parameter gam. In the third level, we optimize the kernel parameter. In the case of the common $'RBF\_kernel'$ the kernel parameter is the bandwidth $sig2$. The evidence of Level 1 is the likelihood of Level 2, and the evidence of Level 2 is the likelihood of Level 3. A clear schematic visualization of this three-level principle is shown in Figure 6.

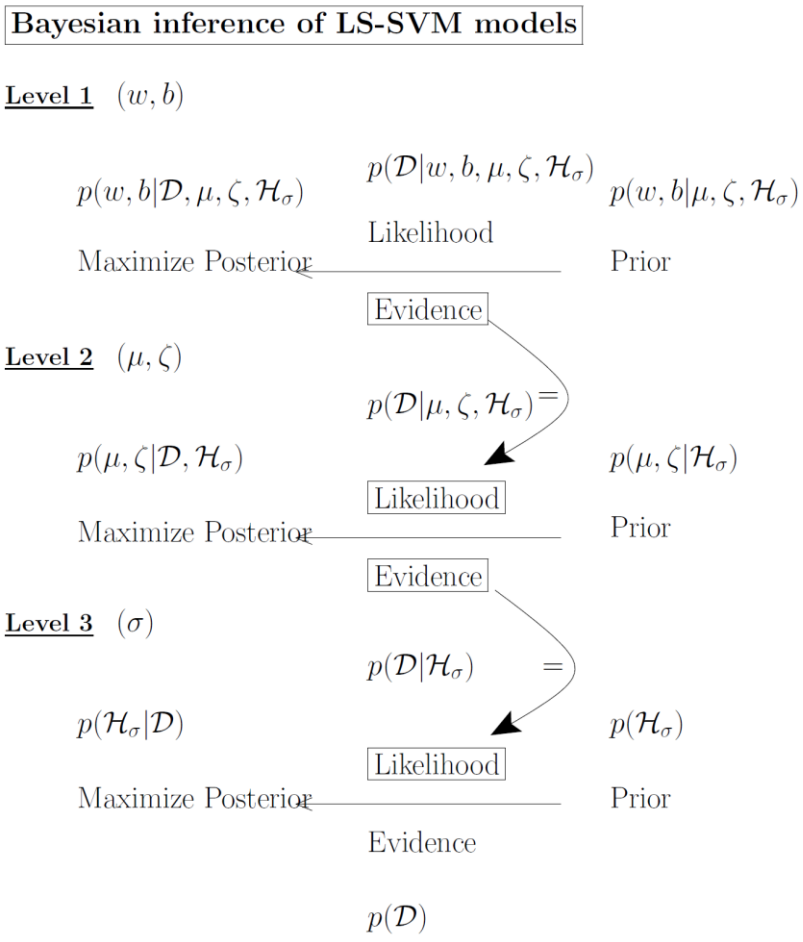


Figure 6: Three-level principle of Bayesian LS-SVM

At first, let $gam = 10, sig2 = 0.05$. Applying the $bay\_optimize$ command for 10 times, there is an optimized model. The error bar is shown in Figure 7.

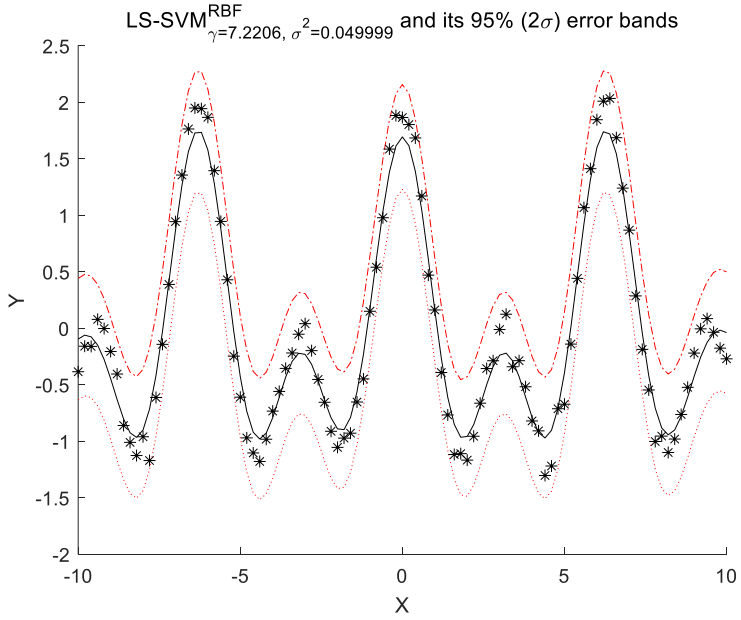


Figure 7: Error bar of Bayesian LS-SVM Regression when optimal

2.4.2 Classification

For classification, it is also possible to get probability estimates. An example using the $Iris$ data-set is shown in Figure 8. The color indicates the probability that the data point (at that position) belongs to the positive class.

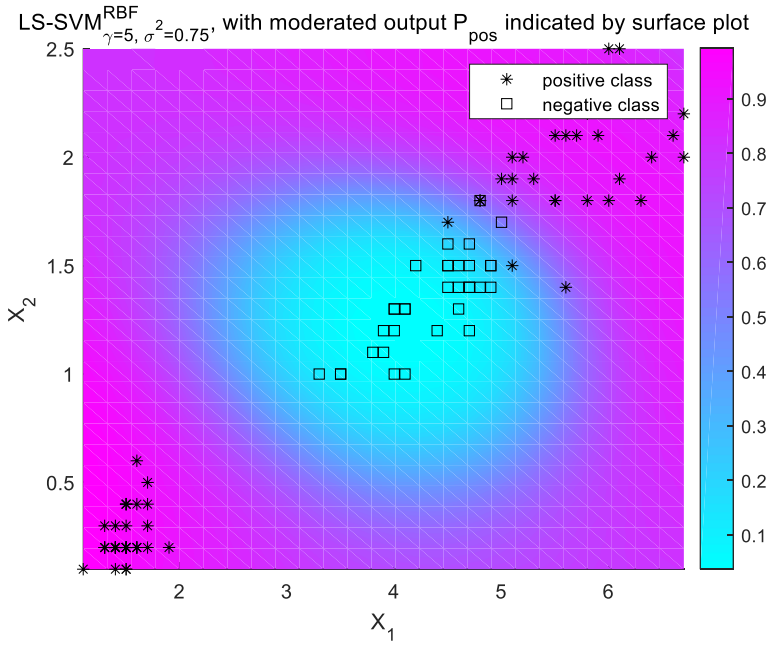
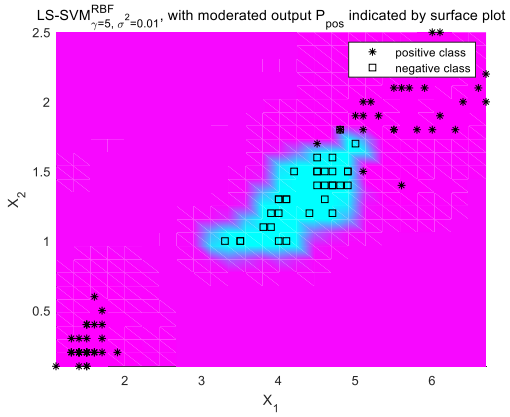


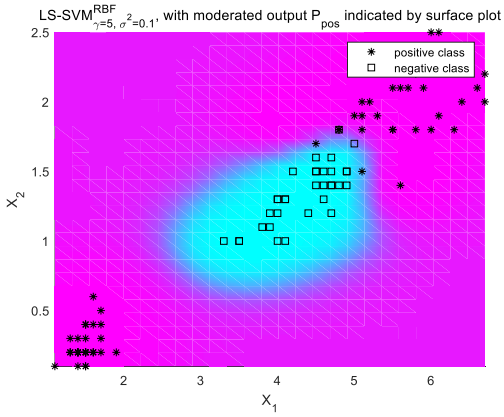
Figure 8: Bayesian LS-SVM Classification for *Iris* dataset

Now let's change the values of gam and $sig2$ and see the influence of given hyper-parameters on this figure.

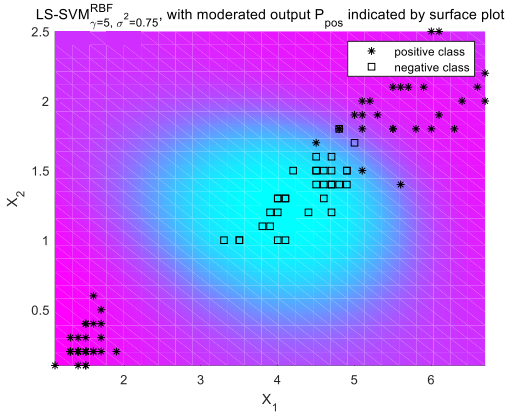
At first , we keep $gam = 5$ and let $sig2$ vary among the range [0.01, 0.1, 0.75, 1, 5, 10]. The results are shown in Figure 9.



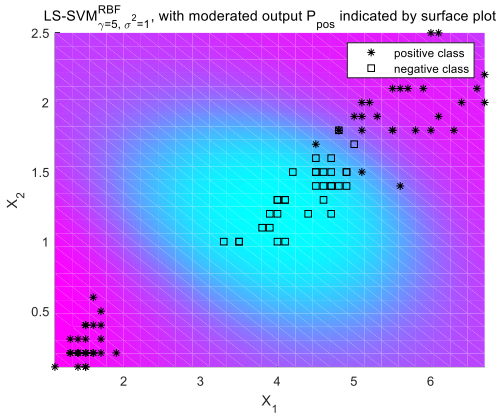
(a) $gam = 5,\ sig2 = 0.01$



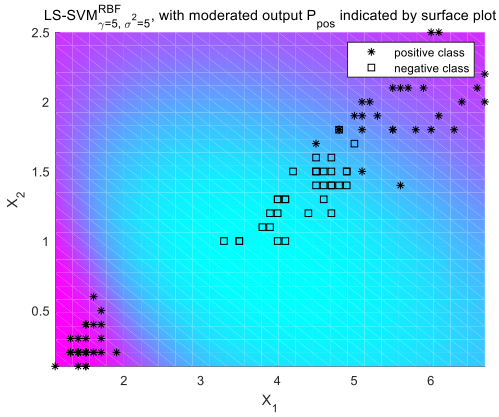
(b) $gam = 5,\ sig2 = 0.1$



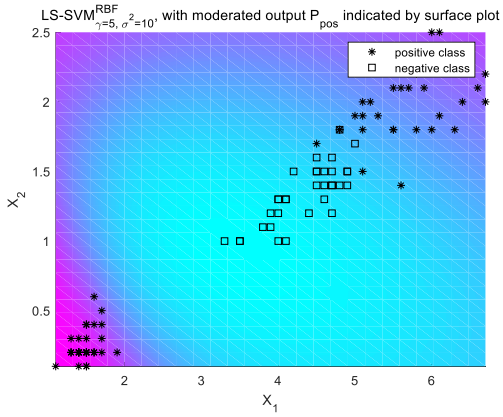
(c) $gam = 5,\ sig2 = 0.75$



(d) $gam = 5,\ sig2 = 1$



(e) $gam = 5,\ sig2 = 5$



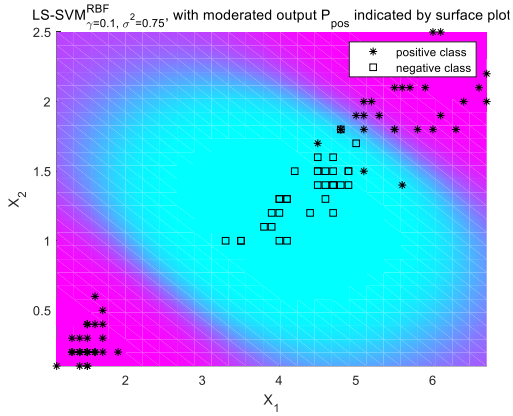
(f) $gam = 5,\ sig2 = 10$

Figure 9: Results with varying *sig2* (*gam*= 5)

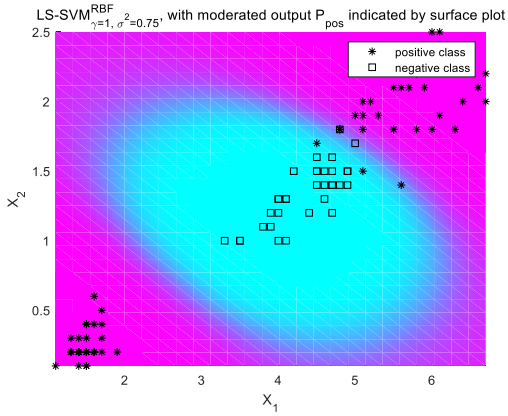
When gam is kept constant, we can see:

- when sigma is large, the decision boundary tends to be linear;
- when sigma is small, the decision boundary becomes highly nonlinear, strict and sharp. It might lead to overfitting.

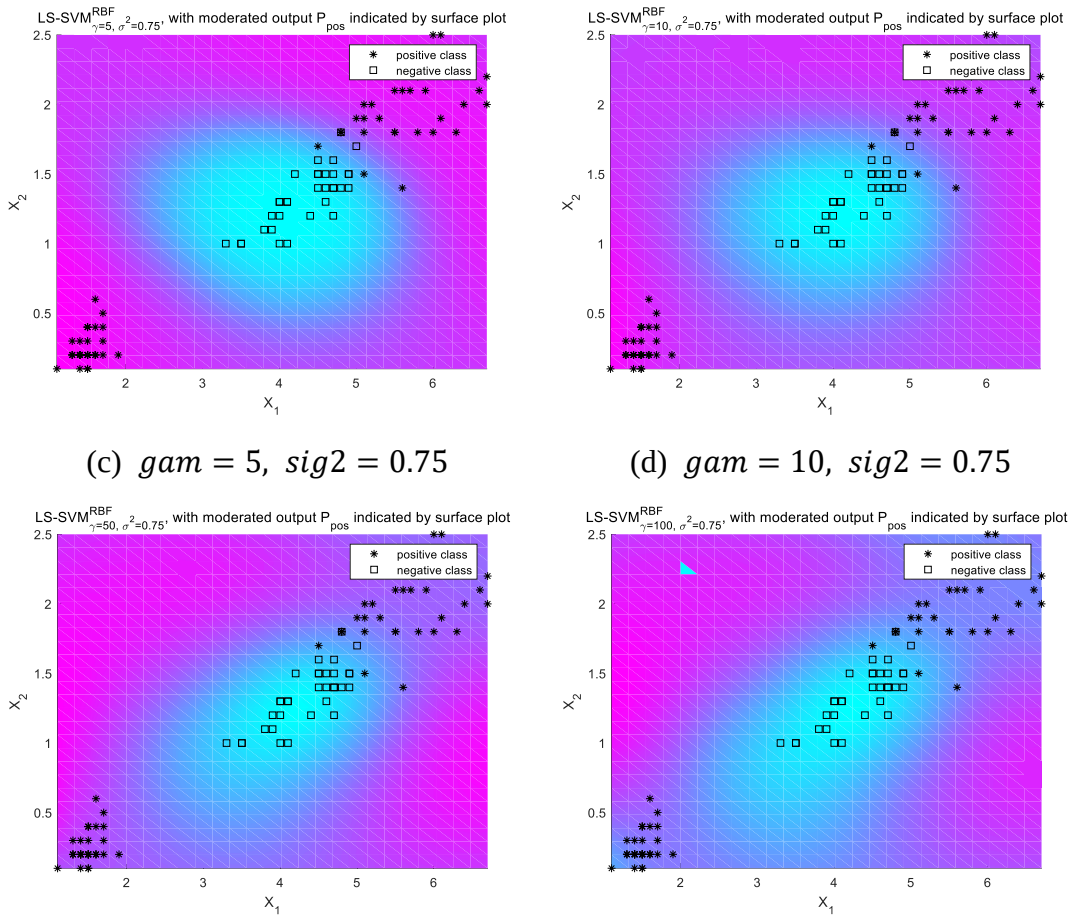
Now we keep $sig2 = 0.75$ and let gam vary among the range [0.1, 1, 5, 10, 50, 100]. The results are shown in Figure 10.



(a) $gam = 0.1,\ sig2 = 0.75$



(b) $gam = 1,\ sig2 = 0.75$



(c) $gam = 5,\ sig2 = 0.75$ (d) $gam = 10,\ sig2 = 0.75$

(e) $gam = 50,\ sig2 = 0.75$ (f) $gam = 100,\ sig2 = 0.75$

Figure 10: Results with varying *gam* (*sig2* = 5)

From Figure 10, when $sig2$ keeps constant (0.75 here), we can see:

- when gam is around the range from 5 to 10, the boundary zone has a round shape. When gam decreases, it becomes elliptic and the long diameter extends to top-left and bottom-right. When gam increases, it also becomes elliptic and the long diameter extends to bottom-left and top-right.
- when gam increases, the blue area shrinks, and the transitional boundary area is larger.

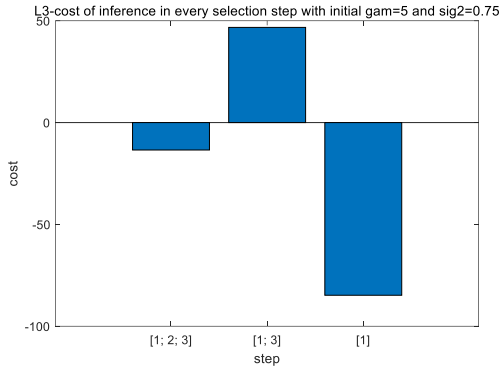
The parameter gam is related to error tolerance. The smaller the gam value, the more certainty the model about the classification. The larger the gam, the model is more focused on classifying every point correctly, therefore, more region with uncertainty left.

2.4.3 Automatic Relevance Determination

The Bayesian framework can also be used to select the most relevant inputs by Automatic Relevance Determination (ARD). For a given problem, one can determine the most relevant inputs for the LS-SVM within the Bayesian evidence framework.

To do so, one assigns a different weighting parameter to each dimension in the kernel and optimizes this using the third level of inference. According to the used kernel, one can remove inputs corresponding the larger or smaller kernel parameters. In this case (a three-dimensional input selection task), it works with the $'RBF\_kernel'$ with a

$sig2$ per input. 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 ($L3 - cost$). Figure 11 visualizes two examples of the result.



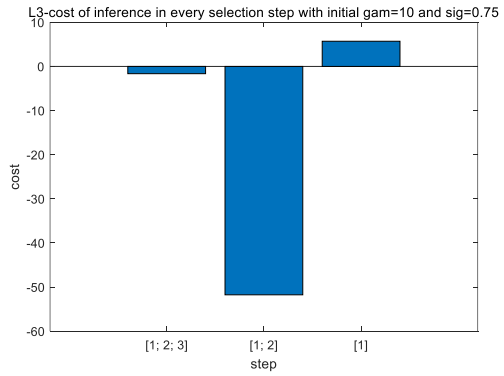
(a)

Step 1: inputs [1: 2: 3].
Input 2 was removed.

Step 2: inputs [1: 3].
Input 3 was removed.

Step 3: input [1].

Lowest $L3 - cost$: at Step 3.
Thus, output dimension: [1].



(b)

Step 1: inputs [1: 2: 3].
Input 3 was removed.

Step 2: inputs [1: 2].
Input 2 was removed.

Step 3: input [1].

Lowest $L3 - cost$: at Step 2.
Thus, output dimension: [1; 2].

Figure 11: Result visualization

To do input selection by using the $crossvalidate$ function instead of the Bayesian framework is also possible. For this dataset, there are 7 types of input dimension(s): [1], [2], [3], [1; 2], [1; 3], [2; 3] and [1; 2; 3]. For each case, we calculate the $cost\_crossval$ according to $crossvalidate$ ($'10 - fold'$). The optimization of the hyper-parameters ($gam, sig2$) is done with the function $tunelssvm$ (with 'csa' + '$simplex$'). The result is shown in Figure 12.

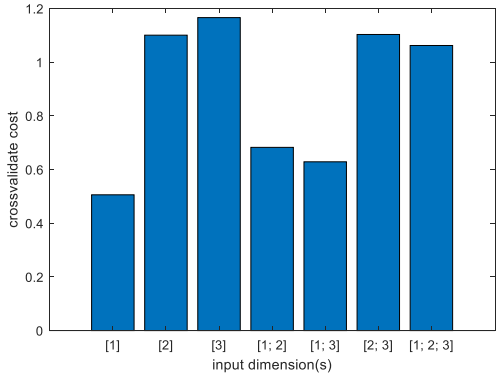
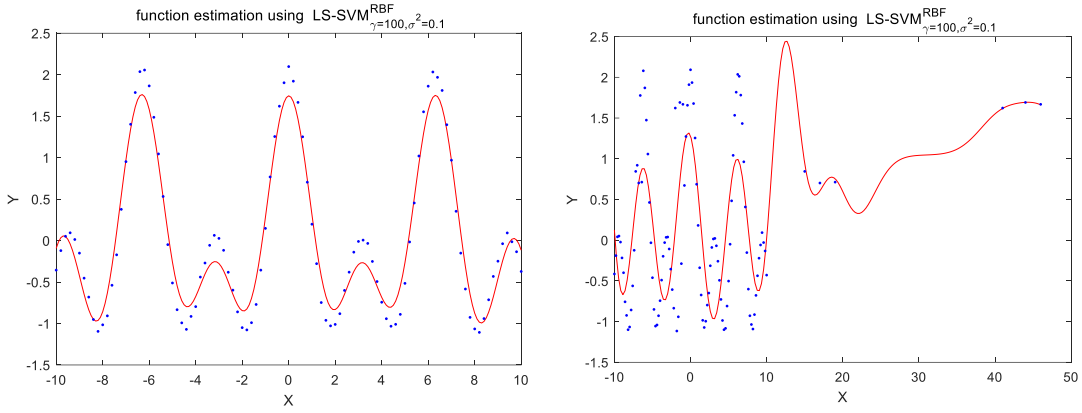


Figure 12: Crossvalidate cost for different input dimensions

From Figure 12, we can also get the same result: input dimension [1].

2.5 Robust Regression

In situations where the data is corrupted with non-Gaussian noise or outliers, it becomes important to incorporate robustness into the estimation. An LS-SVM regression model was applied on a dataset without/with outliers, as shown in Figure 13.



(a) without outliers (b) with outliers

Figure 13: LS-SVM regression without/with outliers

From Figure 13, we can see: in order to fit the outliers, the model is less precise. The weight of the outliers in regression is high.

Now let's train a robust LS-SVM model using the object-oriented interface and robust crossvalidation, as shown in Figure 14.

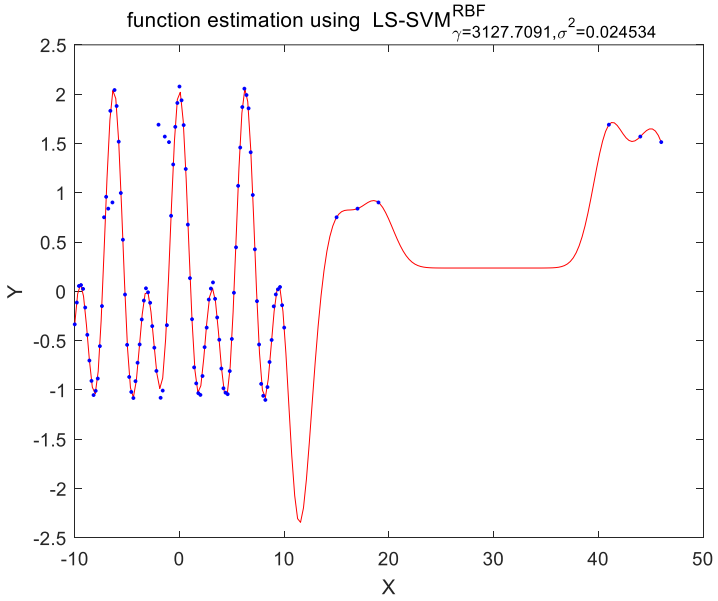
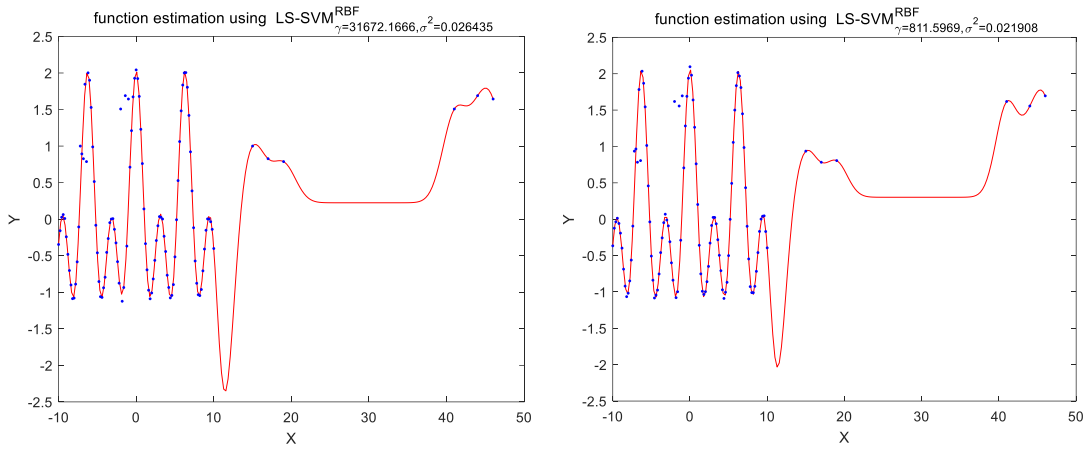


Figure 14: Robust LS-SVM regression

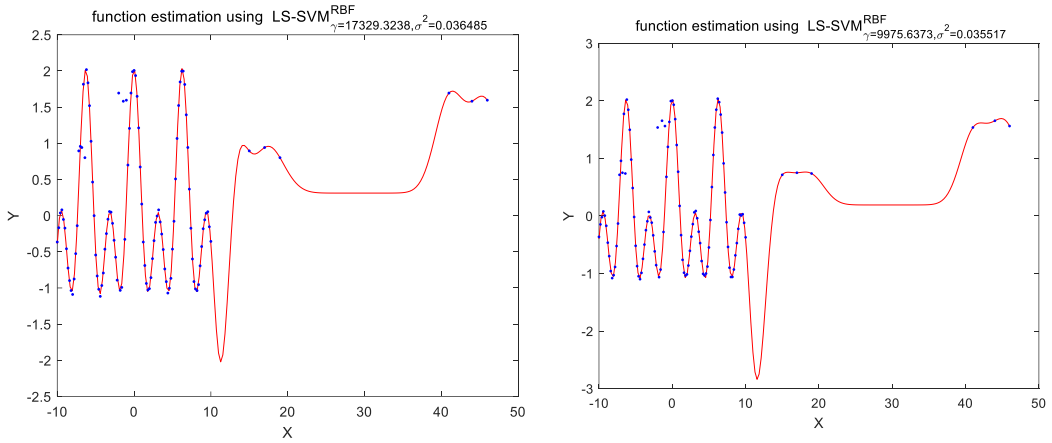
From Figure 14, we can see: the robust regression model has improved a lot compared with the former one. Outliers didn't influence the regression result on those non-outliers

In this case, mean absolute error (mae) is preferred over the classical mean squared error (mse) when we apply $tunelssvm$ function. Otherwise, the errors caused by outliers would be squared and larger, and consequently, put a larger influence on regression.

Try alternatives to the weighting function $wFun$. The result is shown in Figure 15.



(a) *'whuber'* (b) *'whampel'*



(c) *'wlogistic'* (d) *'wmyriad'*

Figure 15: Robust regression with different weighting functions

We can see all these four weighting functions work well.

2.6 Homework Problem

2.6.1 Introduction: Time-series Prediction

A LS-SVM model for time-series prediction was applied on the *logmap* dataset. The result is shown in Figure 16.

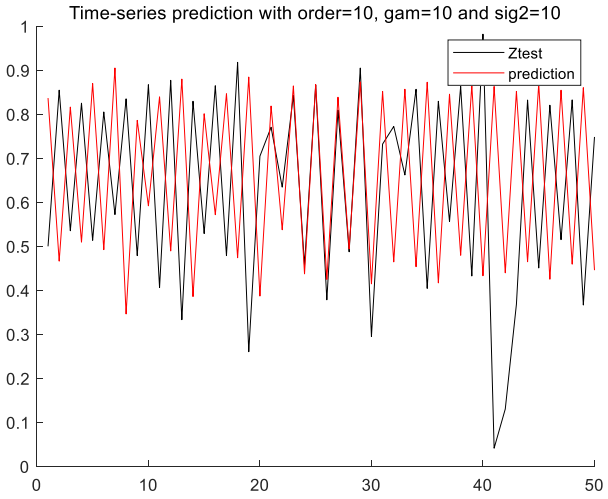


Figure 16: Time-series prediction

According to Figure 16, I feel the performance of this prediction is not good since many peak values and valley values are reverse. We can improve it by optimizing parameters $(gam, sig2)$ and $order$.

Let $order$ vary at the range [2:2:100], and for each $order$, $(gam, sig2)$ pair is determined by the $tunelssvm$ function. For evaluation, mean squared error (MSE) is used. The result is shown in Figure 17.

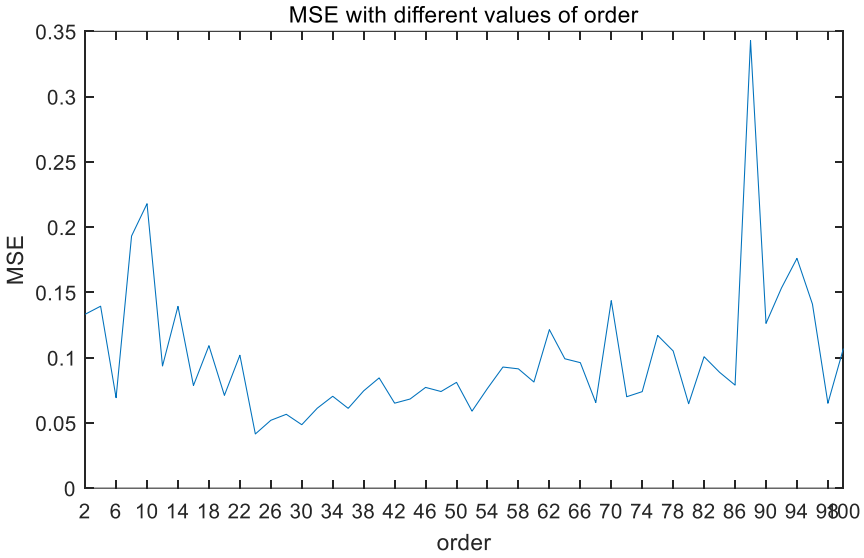


Figure 17: MSE with different values of *order*

According to Figure 17, the smallest MSE occurred when $order = 24$. Generally, when $order$ increases from 2 to 100 with step length 2, MSE decreases first and then increases. Besides, for this dataset, when $order$ increases, MSE varies at a wave shape. It is because the time series reaches peak values and valley values alternatively. When $order$ is 24, $(gam, sig2)$ is (155360.05, 75.583) and MSE is 0.0443.The prediction is shown in Figure 18.

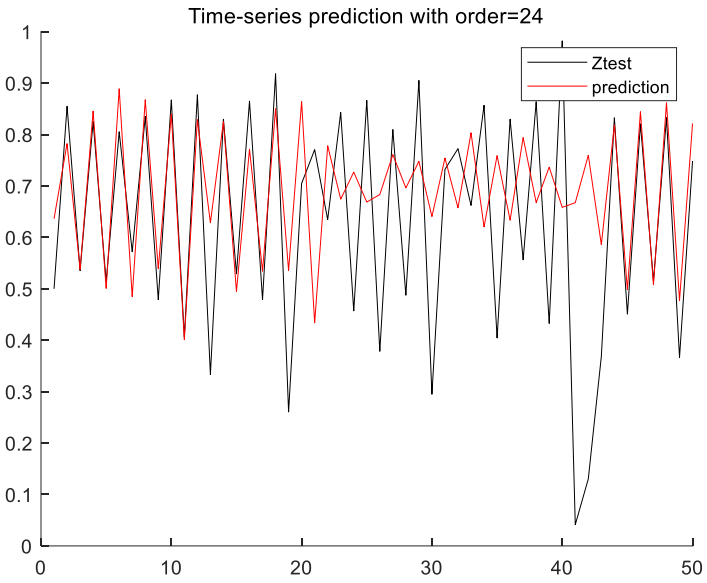


Figure 18: Time series prediction with optimal parameters.

We can see the performance is improved.

2.6.2 Application: Santa Fe Laser Dataset

Now we apply time-series prediction on the Santa Fe Laser dataset. For this utilized auto-regressive model, the prediction result with $order = 50$ is shown in Figure 19. $(gam, sig2)$ is gathered by $tunelssvm$ function.

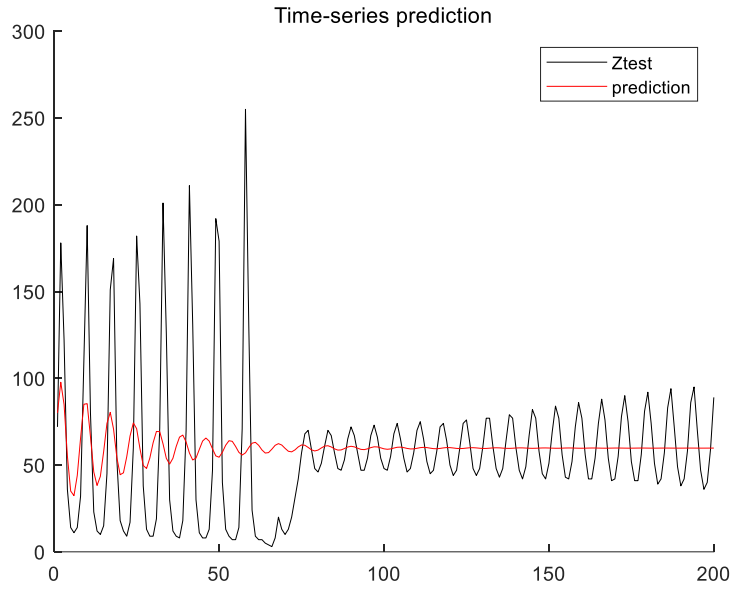


Figure 19: Prediction with *order*=50

According to Figure 19, the performance on the test set is not good. $order = 50$ may not be a good choice.

Let $order$ vary at the range [2:2:100], and for each $order$, $(gam, sig2)$ pair is determined by the $tunelssvm$ function. For evaluation, mean squared error (MSE) is used. The result is shown in Figure 20.

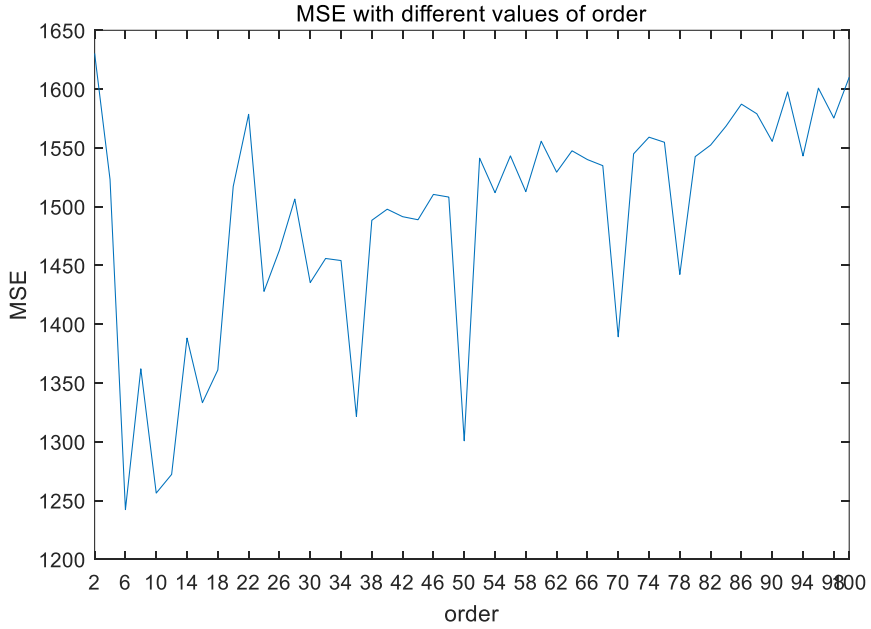


Figure 20: MSE with different values of order

The smallest MSE occurred when $order = 24$ with $MSE = 1321$.The prediction is shown in Figure 21.

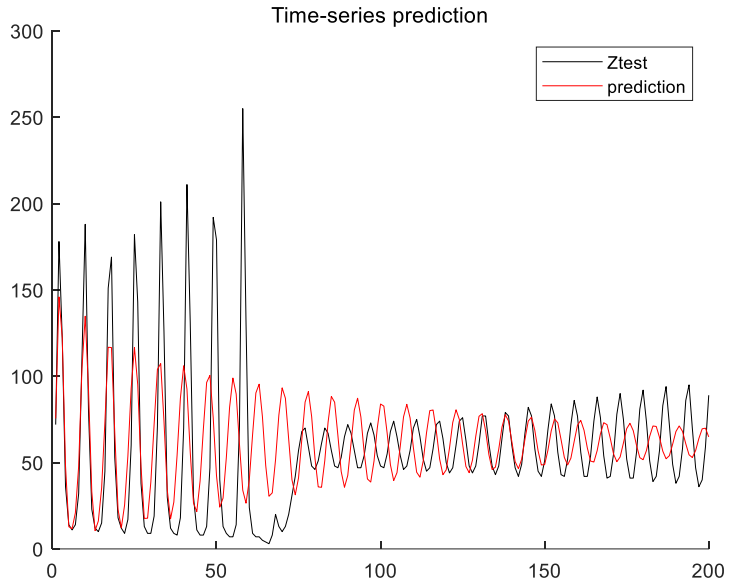


Figure 21: Prediction with *order*=6

The performance is improved.

It would be sensible to use the performance of this recurrent prediction on the validation set to optimize hyper-parameters and the model order. The main reason is that by using validation set, information of test set is not introduced. And when we test models, we can have a more reliable result. However, k-fold cross validation may not be a good choice now. For time series data, we cannot ignore the chronological property. Validation is based on feedforward simulation of the validation set using the feedforwardly trained model. And prediction is done in a recurrent way.