2018 Support Vector Machines
Exam Assignment

Will Steijn
(williamroelof.steijn@student.kuleuven.be)

1 JUNE 2018

KU LEUVEN

1 Exercise Session 1: Classification

1.1 A Simple Example: Two Gaussians

Figure 1: Two Gaussian distributions centered around (1,1) in red and (-1,-1) in blue.

X1 (in red) is a Gaussian distribution centered around (1, 1) and X2 (in blue) is a Gaussian distribution centered around (-1, -1). Both X1 and X2 have the same σ. Since X1 and X2 are overlapping, it is not possible to construct a linear classifier that perfectly classify all the points, meaning some miss-classification is inevitable. The optimal linear classifier in this case (where the two distributions have the same σ) is the line perpendicular to the line that connects the centers for X1 and X2, and is equidistant to the centers for X1 and X2. In this case, the classifier is the $y = -x$ line, with points above this line assigned to X1 and points below the line assigned to X2. Under the conditions where the clouds of points do not have the same σ, the optimal linear classifier will move closer to the cloud with the smaller σ while remaining perpendicular to the line that connects the centers of the two clouds.

1.2 The Support Vector Machine

1. When adding data points to the classes, the classification boundary is generally not dramatically changed when adding data points that are correctly classified, such as a green data point to the area that is already green. However, drastic changes in the position of the classification boundary can occur if, for example, a data point for one class is placed on the wrong side of the classification boundary far from the classification boundary.

2. Adding an outlying data point which lies on the wrong side of the classification boundary has a large affect on the position of the classification hyperplane.

3. The C regularization hyperparameter determines the influence of misclassification on the objective function. The lower the C parameter value, the less the importance placed on avoiding misclassification, which therefore means a separating hyperplane with a wider margin. A higher C paramater value puts more importance on avoiding misclassification, so the separating hyperplane has a smaller margin. This evolution of the width of the margin can be seen when comparing Figure 2 which has a C parameter of 0.01 to Figure 4 which has a C parameter of 100. Figure 2 has many more misclassified points compared to Figure 4. A high value for the C parameter on the training set may lead to overfitting when applied to the test set, so tuning the C parameter is an important consideration when generating the SVM.

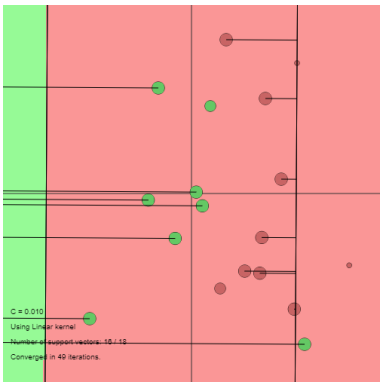


Figure 2: Linear classifier with C parameter = 0.01.

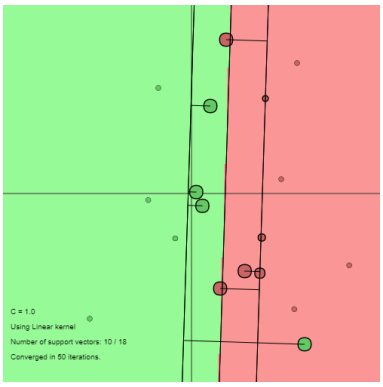


Figure 3: Linear classifier with C parameter = 1.0.

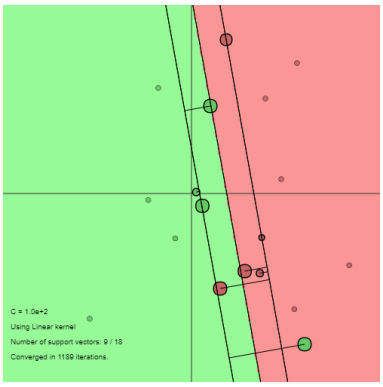


Figure 4: Linear classifier with C parameter = 100.

4.

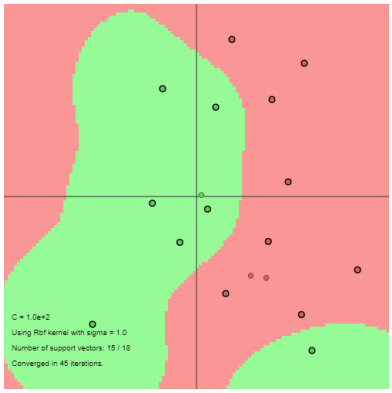


Figure 5: RBF Kernel

Switching back to the RBF kernel leads to perfect classification (Figure 5), which was not possible with a linear kernel due to the data points being linearly inseparable.

5. Small values of the σ hyperparameter seem to yield more flexible boundaries. Figure 6 has the smallest σ, and each green point has its own boundary. As σ increases, the classification boundaries become more general, culminating in Figure 9 where the classifier approaches a linear classifier.

Non-linear classifier with an RBF-kernel with increasing values for the σ parameter and a constant value of 1.0 for the C hyperparameter.

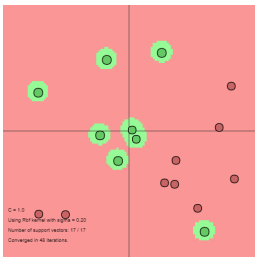


Figure 6: $\sigma = 0.20$

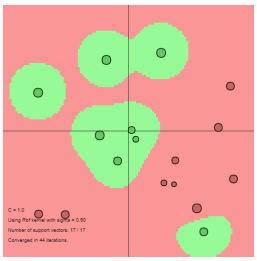


Figure 7: $\sigma = 0.50$

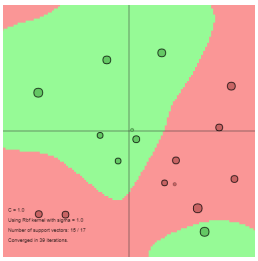


Figure 8: $\sigma = 1.0$

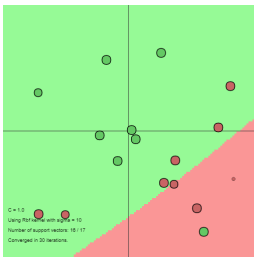


Figure 9: $\sigma = 10$

If the data is almost linearly separable, then the best choice would seem to be a high σ since a highly flexible kernel would probably over-fit the data. A high σ will make the classifier more generalizable, which is advantageous if it will be used on other data. Also, if the data is almost linearly separable then placing a high importance on a few potential misclassifications is not necessary, so a lower value (around 1) for the C hyperparameter would be the best choice. If C is too low (around 0.02), then the classifier seems to classify everything into the group which has a higher number of overall points, which is also not ideal since the data should be approximately separable.

6.

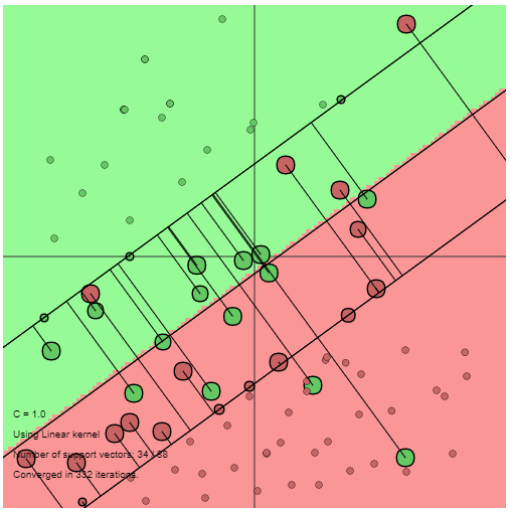


Figure 10: Linear kernel with C = 1.0

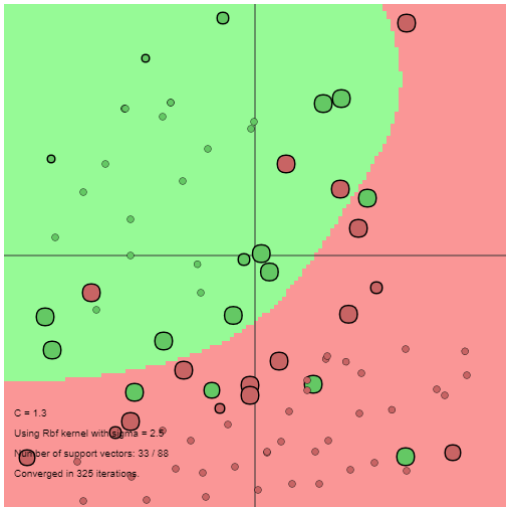


Figure 11: RBF kernel with $\sigma = 2.5$ and C = 1.3.

A linearly non-separable dataset was created where the red points are mostly located in the bottom of the input space and the green points are located in the top left of the input space. Figure 10 shows the result when a SVM with a linear kernel with a C value of 1.0 is applied to this dataset. At this C value, 12 of the 88 points are misclassified and the SVM converged in 332 iterations. Overall, the misclassification rate was pretty stable over the range of C values (except for really small C values where misclassification increased dramatically). Figure 11 shows the result of an SVM with an RBF kernel with $\sigma = 2.5$ and C = 1.3. Since the RBF kernel is non-linear, this means that certain values for the hyperparameters could result in perfect classification. However, perfect classification would almost certainly be overfitting the data, so more conservative parameters were chosen. A σ value of 2.5 and a C values of 1.3 resulted in 8 misclassified points, and gives a curved decision boundary which results in less space classified as green compared to the linear kernel SVM.

7. The support vectors are the data points from which the decision boundary is generated, and therefore lie closest to the decision boundary. Therefore, the decision boundary only depends on these data points and all the rest are unnecessary for classification. With a linear kernel, the support vectors are all the points located within the margin and any points that are outside the margin but are misclassified. A data point that is not a support vector can become a support vector if the C hyperparameter is decreased to the point where this data point is within the margin of the separating hyperplane. The number of support vectors is generally higher for an SVM with an RBF kernel compared to an SVM with a linear kernel because the decision boundaries are more flexible. A data point becomes a support vector for an RBF-SVM if a data point of the opposite class is added that results in the data point becoming the closest point of its class to the decision boundary. In general, decreasing C will result in more support vectors while the minimum number of support vectors is found at a specific σ.

8. The importance of a support vector changes as it becomes more or less important in determining the decision boundary. Points that are misclassified or points that uniquely determine a classification boundary will always have a high importance. Adding support vectors of the same color very close to a support vector with a high importance can lower the importance of that original support vector because the decision boundary in that area is now determined by multiple support vectors instead of just that one.

1.3 Using LS-SVMlab

1.3.1 The Iris Dataset

1. The result of the linear kernel on the Iris dataset is shown in Figure 12. The performance of the linear kernel on the test set was quite poor, as it resulted in misclassification of 11 of the 20 points, giving it a 55% error rate. A second degree polynomial kernel (Figure 13) results in an improved performance on the test set, as it resulted in an error rate of 5%. Both the third degree (Figure 14 and fourth degree (Figure 15) polynomial kernels perfectly classified all the data points in the test set, but the fourth degree polynomial kernel shows signs of overfitting. Therefore, the third degree polynomial kernel is the best choice for this dataset. Increasing the degree of the polynomial increases the flexibility of the classifier, which corresponds

to decreasing the σ hyperparameter of the RBF kernel.

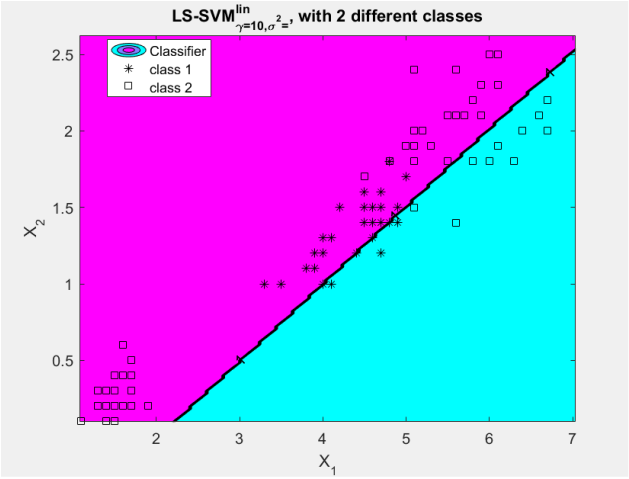


Figure 12: Linear kernel

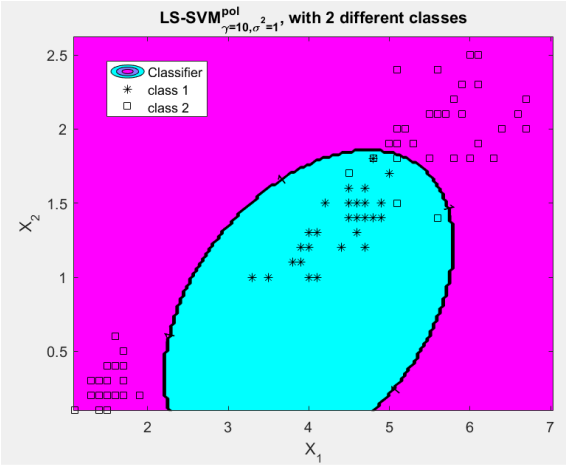


Figure 13: Polynomial kernel with degree = 2

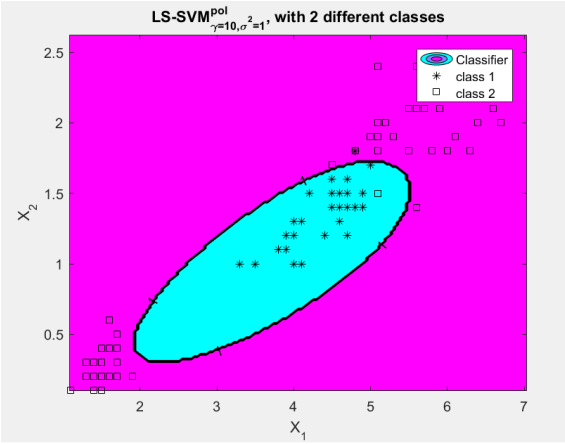


Figure 14: Polynomial kernel with degree = 3

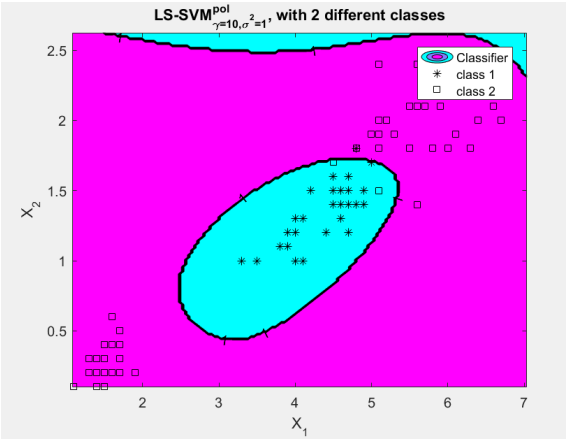


Figure 15: Polynomial kernel with degree = 4

2. Least squares SVMs were generated with an RBF kernel over a range of different σ^2 values and a constant γ value of 10. The error rate for each SVM was calculated on a test set (Figure 16). The lowest σ^2 value tested was 0.01, which resulted in a 10% error rate. The highest error rate occurred at the highest σ^2 values (around 50), which makes sense because previously we have seen high σ^2 leads to a classifier that approximates a linear classifier, and a linear classifier has proven not to work well on this dataset. Classification of the test set was perfect for σ^2 values from about 0.05 to just under 10.

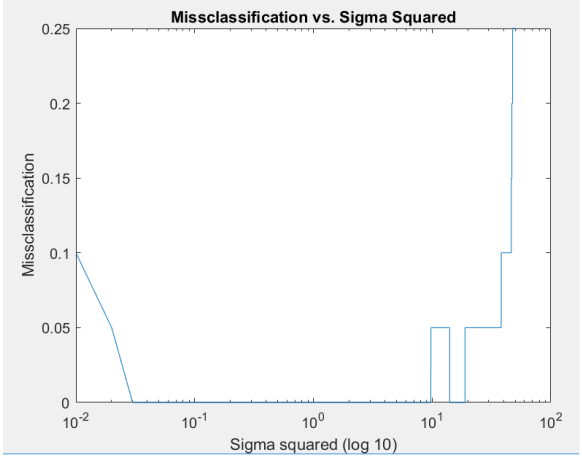


Figure 16: Performance of an LS-SVM with an RBF kernel over a range of σ^2 values.

3. RBF kernel SVMs with a range of γ regularization constants from 0.01 to 50 was tested for performance by calculating the error rate for each SVM on a test set (Figure 17). The σ^2 parameter was set to 1 because this was in the middle of the range of σ^2 values that had an error rate of 0 from the previous question. Figure 17 shows that a good range for γ is from 0.1 to 50, since these values for γ resulted in a 0% misclassification rate on the test set.

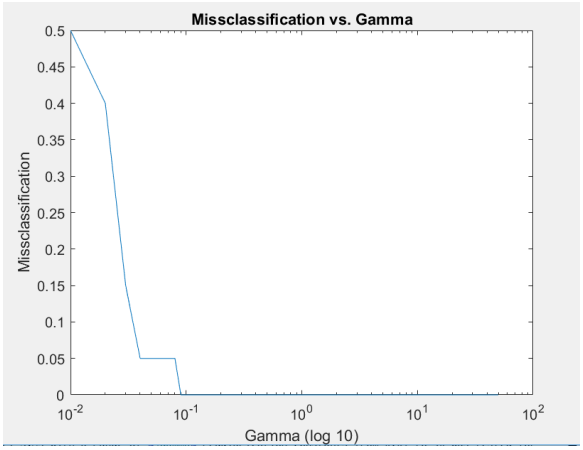


Figure 17: Performance of an LS-SVM with an RBF kernel over a range of γ values.

1.3.2 The Choice of Hyper-parameters

1. The misclassification error on the validation set was tested for all combinations of $\gamma = 1, 5, 10, 50, 100$ and $\sigma^2 = 0.1, 0.5, 1, 10, 50$. As shown in Table 1, the best performance on this split of the data into the training and test set is when γ is large and σ^2 is very small, or when gamma is large and σ^2 is around 10. Both sets these sets of hyper-parameters resulted in no misclassification error.

Table 1: Misclassification error on validation set over different values for γ and σ^2.

		Gamma				
		1	**5**	**10**	**50**	**100**
Sigma	**0.1**	0.05	0.05	0.05	0	0
	0.5	0.05	0.05	0.05	0.05	0.05
	1.0	0.05	0.05	0.05	0.05	0.05
	10	0.05	0.05	0.05	0	0
	50	0.35	0.35	0.30	0	0.05

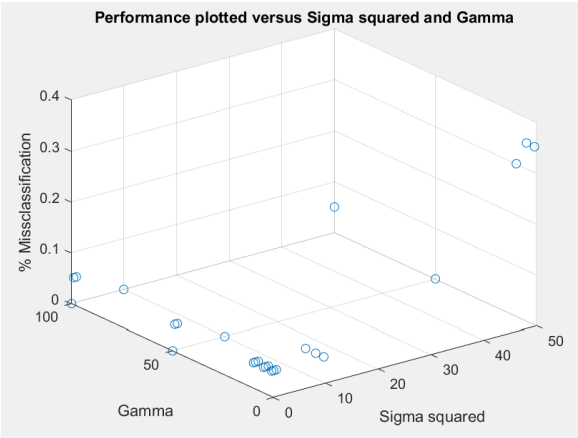


Figure 18: Performance of an LS-SVM with an RBF kernel over a range of γ values and a range of σ^2 values.

The LS-SVM performs the worst when σ^2 is large and γ is small, as its performance on the test set shows an error rate of 35%.

2. Cross-validation using 10 folds is a technique that divides the dataset into 10 equally sized, non-overlapping parts referred to as "folds." Then, the model is trained on the first 9 folds and tested on the 10th fold to get one measurement of error. Next, the model is trained on folds 1-8 and 10, and tested on the 9th fold to get another measurement of error. This application of training and testing is repeated until 10 error measurements are achieved, and then these are averaged together to get the 10-fold cross-validation test error. This can be generalized as k-fold cross-validation, where k is the number of folds the dataset is broken up into. Leave-one-out cross-validation (LOOCV) is when k is equal to the number of data points in the dataset.

Cross-validation is preferred over a simple validation because a simple validation leads to high variance in the test error. When using only one training set and validation set, the chances of getting overly similar or dissimilar training and validation sets increases. k-fold cross validation reduces this effect since the test error is averaged over multiple training and validation splits. Also, using cross-validation results in all the data points being used in generating the model, while in a simple validation the only the specified training data points are used to build the model.

Changing the cross-validation procedure to LOOCV results in an increase in misclassification error on the

Table 2: Comparison of models built using different parameters and different optimization methods for the hyperparameters for the randomly generated dataset from Question 1 of this section.

	Trial 1			Trial 2			Trial 3		
	gam	**sig2**	**cost**	**gam**	**sig2**	**cost**	**gam**	**sig2**	**cost**
csa-simplex	102.58	0.1951	0.0300	0.2865	4.9271	0.0300	0.0531	1.0468	0.0300
csa-grid	1.3792	0.1810	0.0400	65.2469	0.0556	0.0300	253.224	0.0961	0.0300
ds-simplex	5.5191	0.2954	0.0400	1.7386	2.5766	0.0400	1.1108	2.1271	0.0400
ds-grid	0.0485	0.5585	0.0300	0.4025	5.4349	0.0400	0.4086	0.1796	0.0400

dataset generated in the previous question. 10-fold cross-validation results in a 5% misclassification error, while LOOCV results in a misclassification error of 6% ($\gamma = 0.1$, $\sigma^2 = 20$). LOOCV could be preferred if the dataset is small, since it is computationally expensive but also uses the available information as much as possible. However, LOOCV can also lead to overfitting on the training set, while a small k value has similar problems to the simple validation set approach. Therefore, an intermediate k, such as 10, is often chosen.

3. The *tunelssvm* procedure was used to optimize the hyperparameters under different parameters (simplex vs gridsearch) of a model built using different parameters (coupled simulated annealing vs randomized directional search). Table 2 shows the results of different combinations of initial models and optimization functions over three trials. Differences over the three trials among the same model-hyperparameter optimization combinations are a result of different initial conditions leading to different local minima, since neither of these optimization methods guarantee finding the global minimum. Overall, models built with the randomized directional search (ds) method seem to yield more stable results for γ and σ^2 values. However, the csa-built models more consistently lead to finding a lower minima for the cost.

4. Using the training set for evaluating model performance through an ROC curve is not recommended because this will give highly biased results, since the training set is what was used to build the model. The test error for the SVM will be underestimated if an ROC curve is drawn from the training set. The y-axis is the rate of true positive rate (sensitivity) and the x-axis is the false positive rate (1-specificity). Figure 19 shows an ROC curve for the RBF LS-SVM with a set of optimized hyperparameters found in the previous question ($\gamma = .0485$, $\sigma^2 = .5585$). The area under the curve is 1, which means perfect classification performance on the validation set. The area under the curve is much less for the ROC curve from the LS-SVM with hyperparameters known to lead to poor performance (Figure 20).

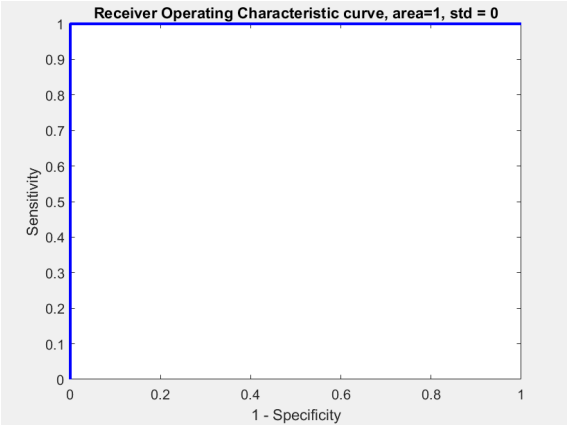


Figure 19: ROC curve for LS-SVM trained on the training data from question 1 of this section with $\gamma = .0485$, $\sigma^2 = .5585$.

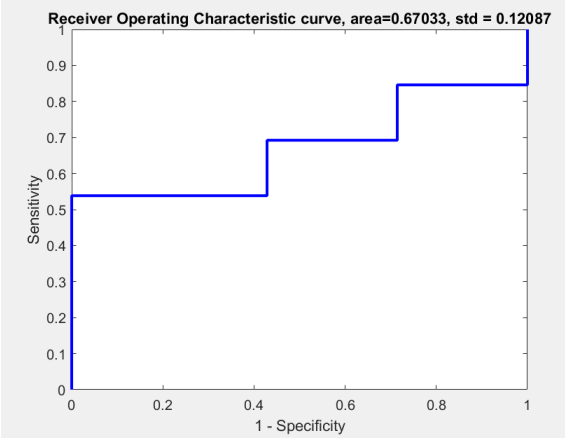


Figure 20: ROC curve for LS-SVM trained on the training data from question 1 of this section with $\gamma = 1$, $\sigma^2 = 50$.

1.4 Homework

1.4.1 The Ripley Dataset

1. Data exploration of the plotted training (Figure 21) and test (Figure 22) sets shows that the two classes are slightly overlapping and each class is made up of two subpopulations, one below 0 and one above 0 on the X1 axis. It appears that a linear classifier X2 = 0.5 might be reasonable for this data, although it would struggle between X2 = 0.4 and X2 = 0.6 where much of the overlapping occurs.

Figure 21: Scatterplot of the training data from the Ripley dataset.

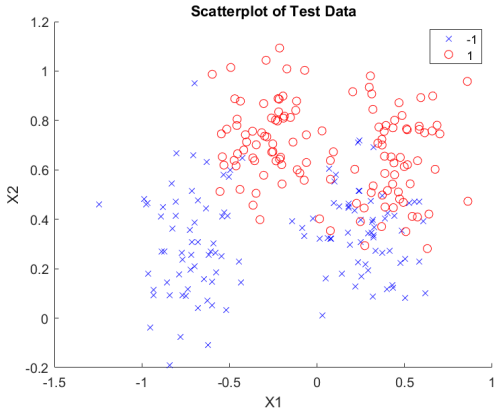


Figure 22: Scatterplot of the test data from the Ripley dataset.

2. A LS-SVM with a linear kernel was trained on 750 data points that made up the training set, and the resulting classifier can be seen in Figure 23. The γ parameter was optimized using the *tunelssvm* function with a csa-built linear model and a simplex optimization procedure, resulting in a γ value of 0.0012. The

linear model performed fairly well, as it had an error rate of 13.60% when applied to the test set of 250 data points, and an area under the curve of 0.93587 in the ROC plot (Figure 24).

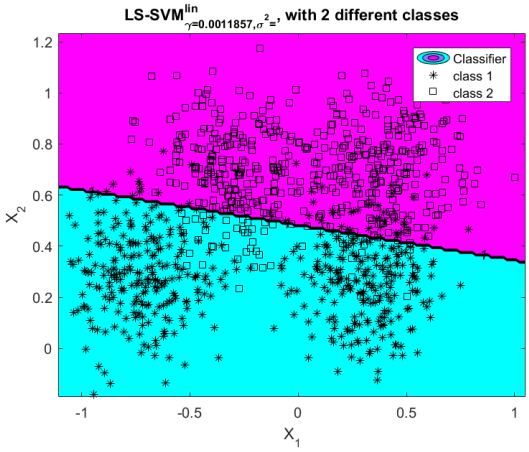


Figure 23: LS-SVM with a linear kernel on the ripley dataset.

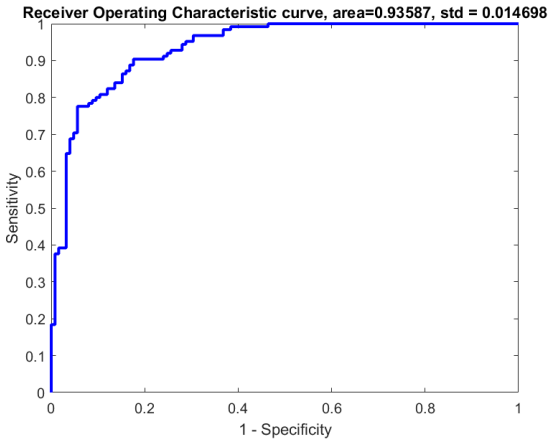


Figure 24: ROC curve for LS-SVM with a linear kernel on the ripley dataset.

3. Next, an LS-SVM with an RBF kernel was generated on the training set and the γ and σ^2 parameters were tuned again using *tunelssvm*. Multiple tuning iterations were done using the csa model-building option and the simplex optimization option. The tuned parameters, particularly σ^2, did not change dramatically over 4 of the 5 iterations. However, the 5th iteration resulted in a γ of nearly 20,000 while for all the other iterations, γ was between 0.4 and 1.5. Regardless of the variation in parameters, the performance was very stable, as the lowest cost was 0.077 and the highest was 0.081. The parameters that gave the lowest cost were chosen ($\gamma = 0.4292$, $\sigma^2 = 0.4168$). The result of the RBF kernel model on the training data is shown in Figure 25. When applied to the test set, the RBF kernel model resulted in an error rate of 13.20%.

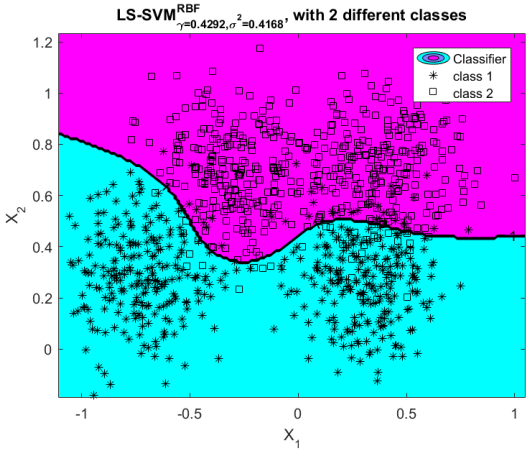


Figure 25: LS-SVM with an RBF kernel on the ripley dataset.

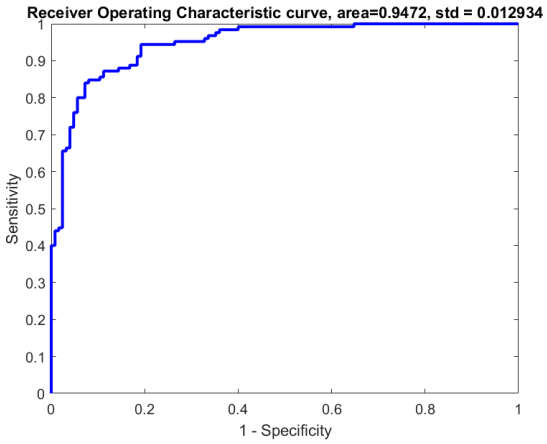


Figure 26: ROC curve for LS-SVM with an RBF kernel on the ripley dataset.

4. The ROC curve for the RBF kernel is shown in Figure 26, and the area under the curve was 0.9472. This is slightly higher than the area under the curve for the linear kernel model, which, along with the slightly lower error rate on the test set, suggests the RBF kernel is slightly better than a linear kernel.

5. The final model chosen is the LS-SVM with an RBF kernel and a $\gamma = 0.4292$ and $\sigma^2 = 0.4168$. The RBF kernel was chosen over the linear kernel because it lead to a lower 10-fold crossvalidation misclassification error on the training set, a lower misclassification error on the test error, and a larger area under the ROC curve. The increase in performance of the RBF kernel over the linear kernel makes up for the increased difficulty of interpretability of the more complex RBF model. The RBF kernel is not perfectly suited for this dataset because there is still some misclassification error, but due to all the overlapping of the classes, a perfect classifier is not possible without extreme over-fitting.

1.4.2 The Breast Cancer Dataset

1. The breast cancer dataset is more complex than the previous datasets, as this dataset has 30 explanatory variables. Due to this high dimensionality, a simple scatterplot is not sufficient to completely visualize the data. However, some combinations of explanatory variables show that the two classes cluster separately over these two variables (Figures 27,28, 29).

2. A tuned linear kernel resulted in a γ of 0.1216. The linear kernel with this value for γ resulted in an LS-SVM that correctly classified 95.27% of the 169 data points from the test set.

Figure 27: Scatterplot of first two variables from the breast cancer dataset and the corresponding class.

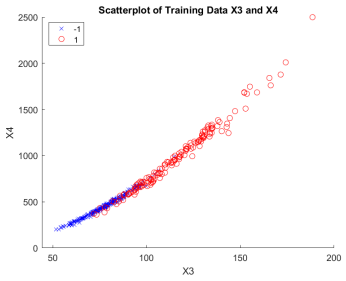


Figure 28: Scatterplot of the third and fourth variables from the breast cancer dataset and the corresponding class.

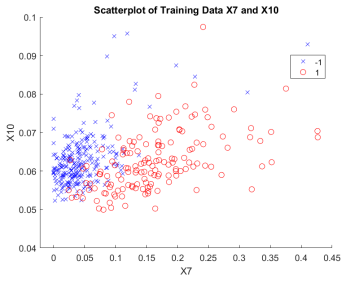


Figure 29: Scatterplot of the seventh and tenth variables from the breast cancer dataset and the corresponding class.

3. An RBF kernel was tuned multiple times using csa for model building and simplex for optimization. The parameters were relatively stable over multiple tuning runs, with every tuning leading to a large γ (from 28 to 125) and a large σ^2 (from 28 to 164). The cost ranged from 0.01 to 0.02 over the multiple iterations. The parameters that resulted in the lowest cost ($\gamma = 37.8640$, $\sigma^2 = 31.9563$) were used in the final RBF kernel model. This model correctly classified 97.63% of the data points in the test set, which was a slightly better performance than the linear kernel.

4. The area under the ROC curve for the model with a linear kernel is greater than the area under the curve for the RBF kernel (Figures 30, 31). This suggests a linear model is preferred over an RBF kernel.

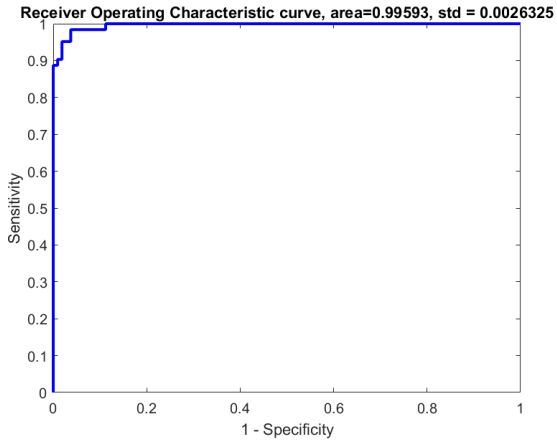


Figure 30: ROC curve for LS-SVM with a linear kernel on the breast cancer dataset.

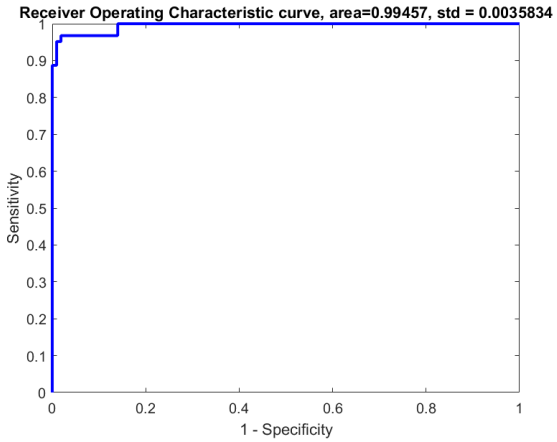


Figure 31: ROC curve for LS-SVM with an RBF kernel on the breast cancer dataset.

5. The linear model is chosen as the final model due because of lack of clear evidence that the RBF kernel is better. The RBF kernel had a better performance on the test set, but had a smaller area under the ROC curve. Therefore the simpler model should be chosen, which is the linear model. Furthermore, a large σ^2

value with a large regularization parameter, such as was the case with the RBF model chosen for this dataset, leads to the RBF kernel approximating a linear model. This gives move evidence that the best classifier for this dataset is linear. Overall, the misclassification error for the linear model on the test set is very small, and the area under the ROC curve for the linear model is very close to one. This suggests that the methodology used above is highly suitable for this dataset.

1.4.3 Diabetes Database

1. The Diabetes data set has 8 explanatory variables. The training set has 300 observations and the test set consists of 168 observations. Since this data set is also of a higher dimensionality than one that allows for simple visualization, different combinations of explanatory variables were plotted to explore the separability of the two classes of data points. This two classes seem to be highly mixed, since there does not seem to be any combination of variables that shows a clear separability of the classes. Figures 32, 33, and 34 shows that there appears to be a lot of overlap between classes.

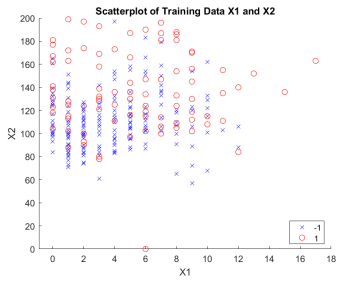


Figure 32: Scatterplot of first two variables from the diabetes dataset and the corresponding class.

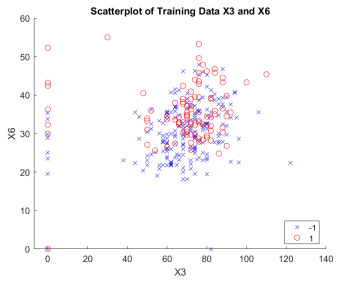


Figure 33: Scatterplot of the third and sixth variables from the diabetes dataset and the corresponding class.

Figure 34: Scatterplot of the fourth and eighth variables from the diabetes dataset and the corresponding class.

2. A tuned linear kernel resulted in a γ of 0.0665. The linear kernel with this value for γ resulted in an LS-SVM that correctly classified 79.76% of the 168 data points from the test set. This performance is better than expected based off the data exploration from the previous question, but indeed indeed is worse than the models for the previous two datasets, which was expected.

3. An RBF kernel was tuned multiple times using csa for model building and simplex for optimization, which consistently resulted in a very large σ^2 and a performance of around 25% misclassification. The γ parameter was a little more varied over the multiple iterations, with a low of 0.6 and a high of 749. The final parameters used for the generation of the RBF kernel model were a γ of 3.86 and a σ^2 of 228.71. This large σ^2 means the RBF kernel is approximating a linear model. The RBF model correctly classified 77.98% of the data points from the test set, which is slightly worse than the linear classifier.

4. The area under the ROC curve for the model with an RBF kernel is greater than the area under the curve for the linear kernel (0.844 vs 0.848). However, this difference is too small to make any conclusions about

which model is better.

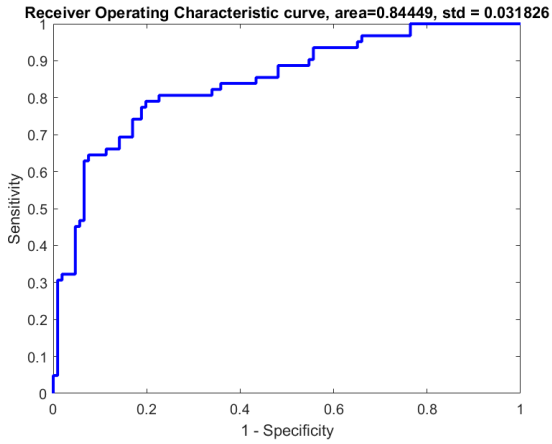


Figure 35: ROC curve for LS-SVM with a linear kernel on the diabetes dataset.

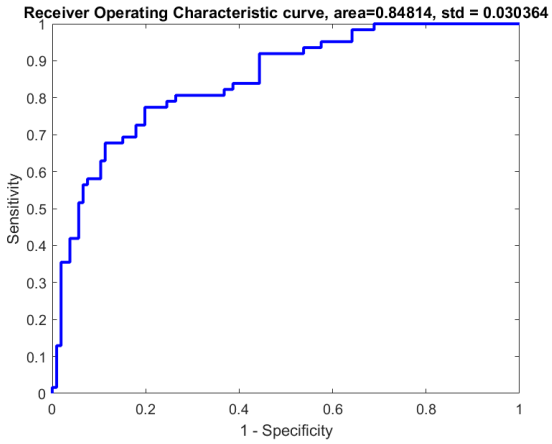


Figure 36: ROC curve for LS-SVM with an RBF kernel on the diabetes dataset.

5. The final model chosen for this dataset is the linear model, again due to the preference for simplicity when there is a lack of a clear difference in performance between the RBF kernel and the linear model. However, unlike with the breast cancer dataset, the linear model did not perform that well overall as it misclassified slightly over 20% of the data points in the test set. This suggests that this methodology might not be best suited for this dataset. However, there appears to be significant overlap of the classes in this dataset so it is possible that a better methodology for classification does not exist.

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

2.1 The Support Vector Machine for Regression

A random dataset of 20 data points was constructed and linear kernel regression was compared to RBF kernel regression. The linear kernel with a bound of 20 and e=0.5 performed poorly, due to the highly non-linear structure of the data (Figure 37). The high flexibility of the RBF kernel with low σ allowed it to perform much better than the linear kernel on this dataset (Figure 38).

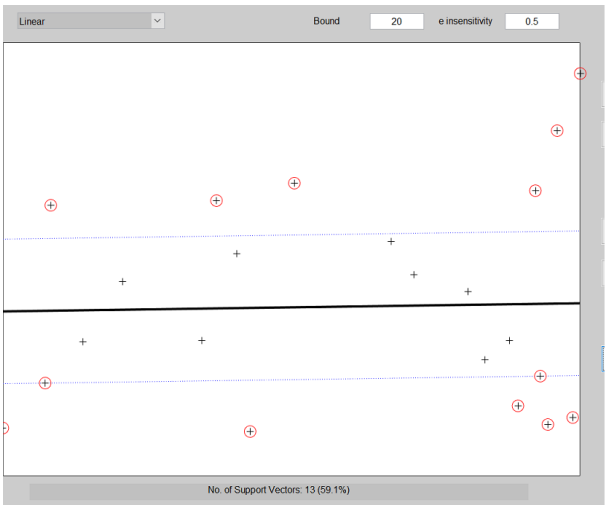


Figure 37: Linear kernel regression applied to a random dataset.

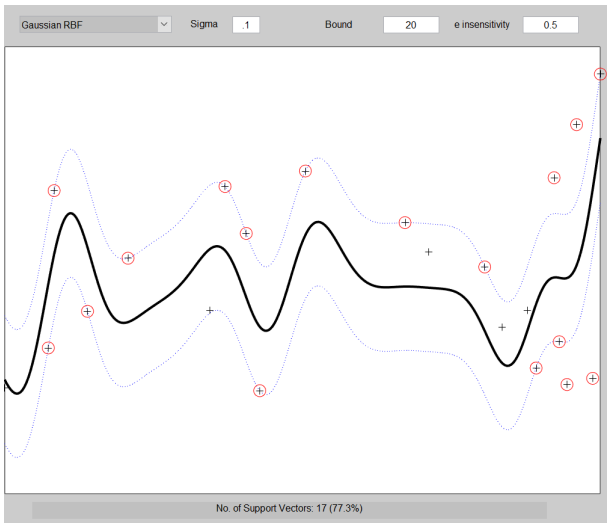


Figure 38: RBF kernel regression applied to a random dataset.

A dataset was constructed in the *uiregress* function of 26 data points where a linear kernel fits better than any other kernel. The dataset is shown in Figure 39 with a linear kernel regression line with an infinite bound and e of 0, which corresponds to a classical least squares fit.

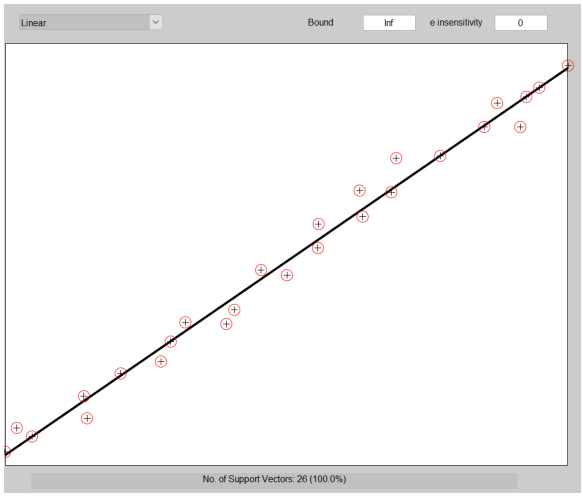
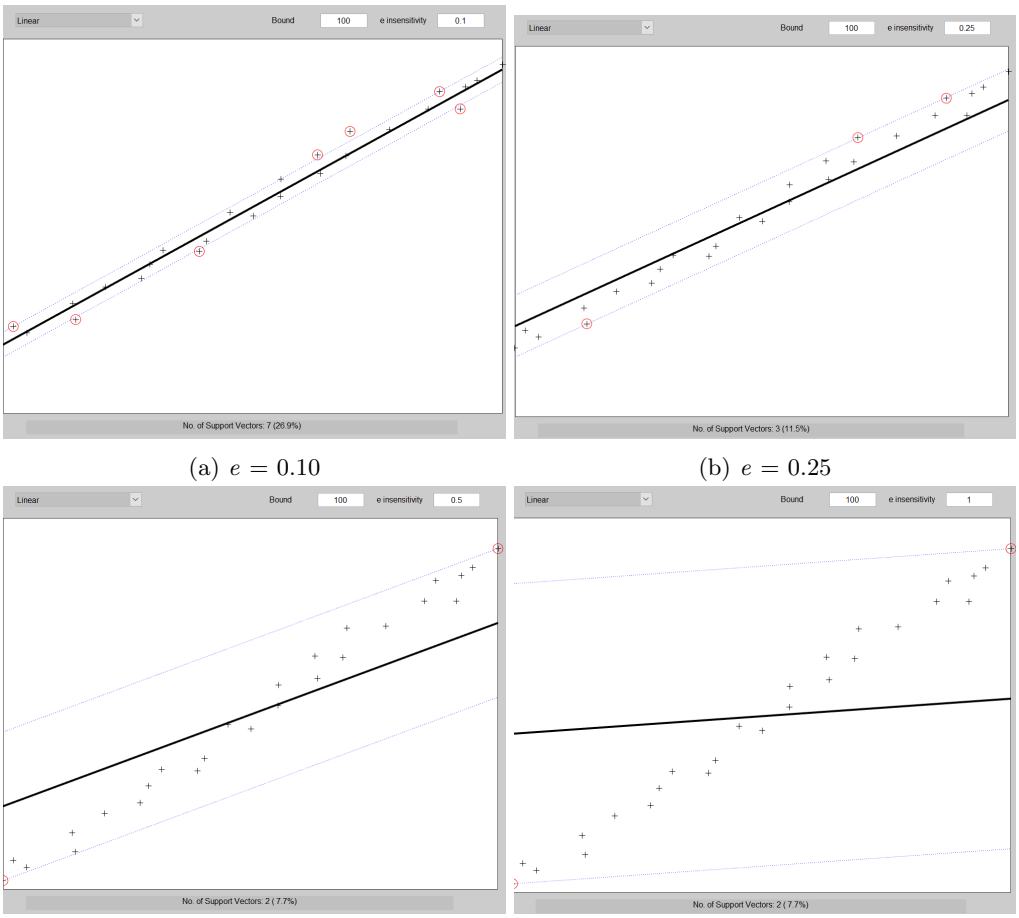


Figure 39: Linear kernel regression with infinite bound and e=0.

The influence of different values of e was investigated by examining the fit of linear kernels with the bound set to 100 and e values set to 0.1, 0.25, 0.5, and 1 (Figure 40).

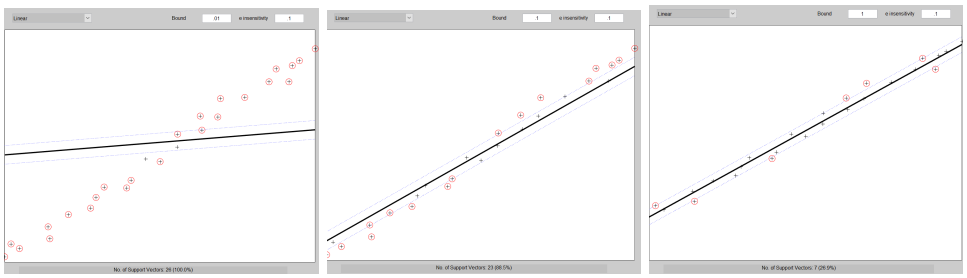


(a) $e = 0.10$ (b) $e = 0.25$

(c) $e = 0.50$ (d) $e = 1.0$

Figure 40: Linear kernel regression with different values for the ϵ-sensitive loss function.

The e value is the ϵ-sensitive loss function, which determines the accuracy of the function approximation. As e increases the required accuracy decreases, meaning the number of support vectors decreases and the margin increases. Next, the influence of the bound parameter was investigated by examining the fit of linear kernels with e set to 0.1 and the bound set to 0.01, 0.10, and 1.00 (Figure 41). The bound is like the c parameter discussed in the previous exercise session, as it determines how deviations larger than the desired accuracy are tolerated. Therefore, a large bound means increased importance for finding the regression line that minimizes the error for each data point, when e is small, while a smaller bound will give a flatter regression line.



(a) bound = 0.01 (b) bound = 0.10 (c) bound = 1

Figure 41: Linear kernel regression with different values for the bound parameter.

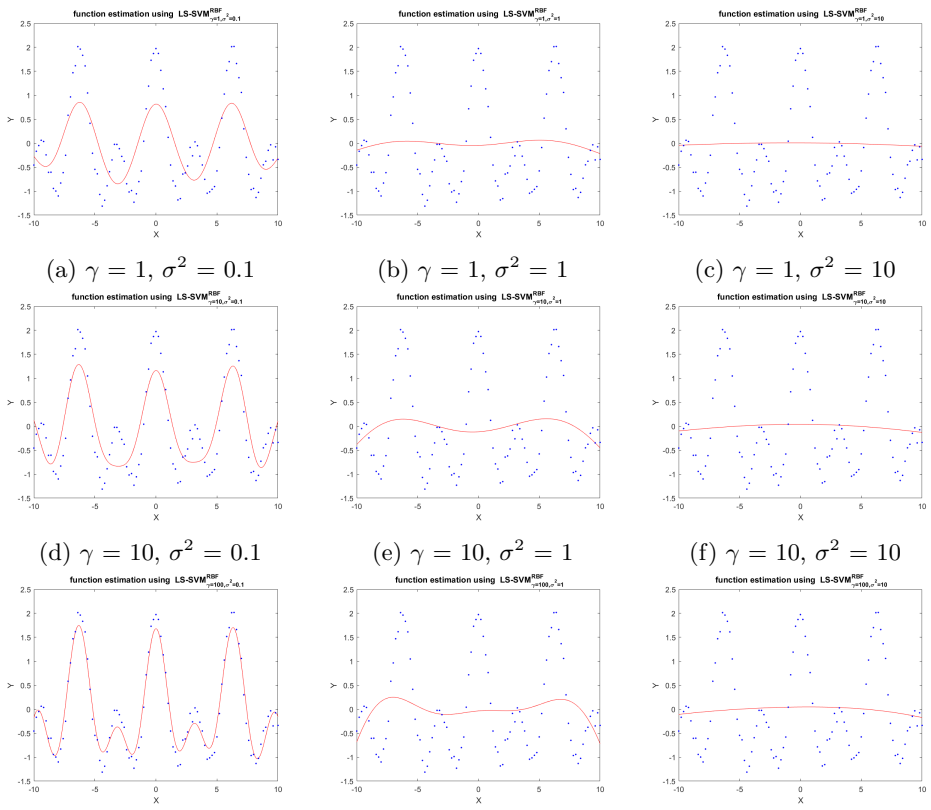
As the bound increases, the number of support vectors decreases and the accuracy gets closer to the e value. The sparsity property comes into play as e increases, and the support value of the data points goes to 0 (the number of support vectors decreases). SVM regression is different from a classical least squares fit in that with SVM regression you can set the accuracy and amount of deviation from that error you tolerate, while a classical least squares fit has e set to 0.

2.2 A Simple Example: Sum of Cosines

Fucntion estimation with an LSSVM with an RBF kernel was tested on a generated dataset. The dataset was split up into a training and test set, and different values for γ and σ^2 were evaluated. The model was generated from the training set. The summed squared error is found by adding the squared difference between the predicted Y values and the actual Y values of the test set. The models' fit on the training data is show in Figure 42 , and on the test data in Figure 43. It appears that the fit improves as σ^2 decreases and γ increases. This is confirmed by the squared error of the models on the test set, which is lowest when $\sigma^2 = 0.1$ and $\gamma = 100$ (Table 3).

Table 3: Squared error for RBF kernel LSSVM with different settings for the hyperparameters.

		gam		
		1	**10**	**100**
Sig2	**0.1**	45.5325	26.6721	5.289
	1	91.7937	90.1602	86.8973
	10	93.0538	92.8387	92.7226



(a) $\gamma = 1$, $\sigma^2 = 0.1$ (b) $\gamma = 1$, $\sigma^2 = 1$ (c) $\gamma = 1$, $\sigma^2 = 10$

(d) $\gamma = 10$, $\sigma^2 = 0.1$ (e) $\gamma = 10$, $\sigma^2 = 1$ (f) $\gamma = 10$, $\sigma^2 = 10$

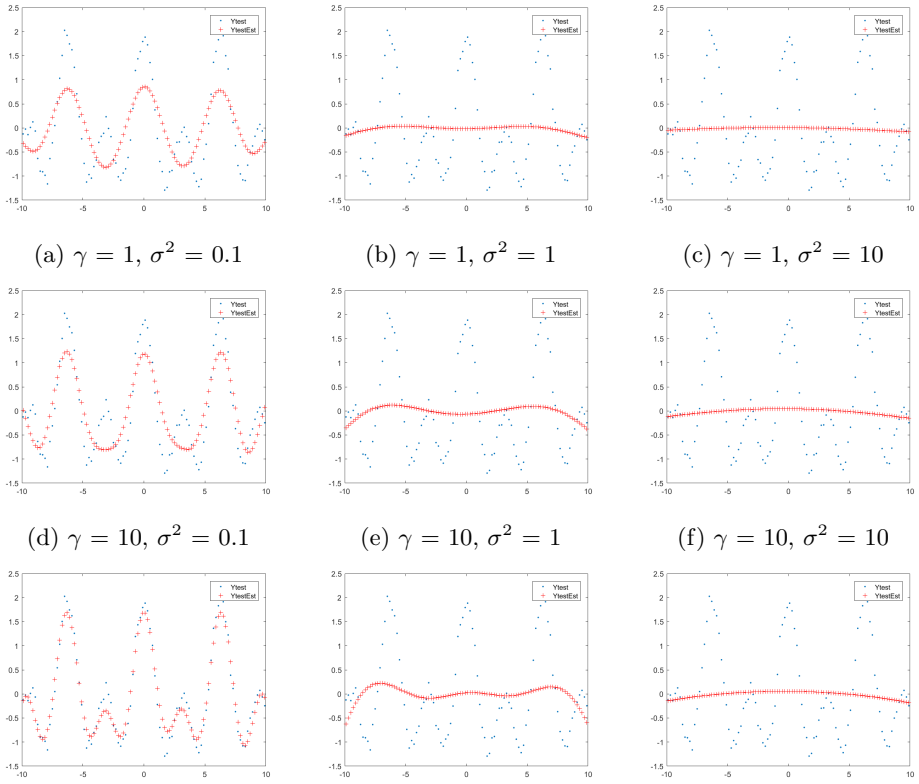
(g) $\gamma = 100$, $\sigma^2 = 0.1$ (h) $\gamma = 100$, $\sigma^2 = 1$ (i) $\gamma = 100$, $\sigma^2 = 10$

Figure 42: Function estimation on the training set for RBF kernel LSSVM with different combinations of parameters.

To find the the optimal pair of hyperparameters, combinations of smaller values for σ^2 and larger values for γ were investigated (Table 4). The minimum squared error occurred when σ^2 was equal to 0.1 and γ was equal to 10,000 or larger. I do not think there is one specific pair of optimal hyperparameters. The value for σ^2 has the largest effect on model performance as the LSSVM seems to perform equally well across a range of high γ values when σ^2 was equal to 0.1.

Table 4: Squared error for RBF kernel LSSVM with different settings for the hyperparameters.

		gam			
		100	**1000**	**10000**	**100000**
Sig2	**0.1**	5.6736	1.6525	1.4873	1.4728
	0.01	1.5836	1.6939	1.8383	1.8878
	0.001	2.0425	2.0887	2.0939	2.0944
	0.0001	27.7896	27.3894	27.3491	27.3451



(a) $\gamma = 1$, $\sigma^2 = 0.1$ (b) $\gamma = 1$, $\sigma^2 = 1$ (c) $\gamma = 1$, $\sigma^2 = 10$

(d) $\gamma = 10$, $\sigma^2 = 0.1$ (e) $\gamma = 10$, $\sigma^2 = 1$ (f) $\gamma = 10$, $\sigma^2 = 10$

(g) $\gamma = 100$, $\sigma^2 = 0.1$ (h) $\gamma = 100$, $\sigma^2 = 1$ (i) $\gamma = 100$, $\sigma^2 = 10$

Figure 43: Function estimation on the test set for RBF kernel LSSVM with different combinations of parameters.

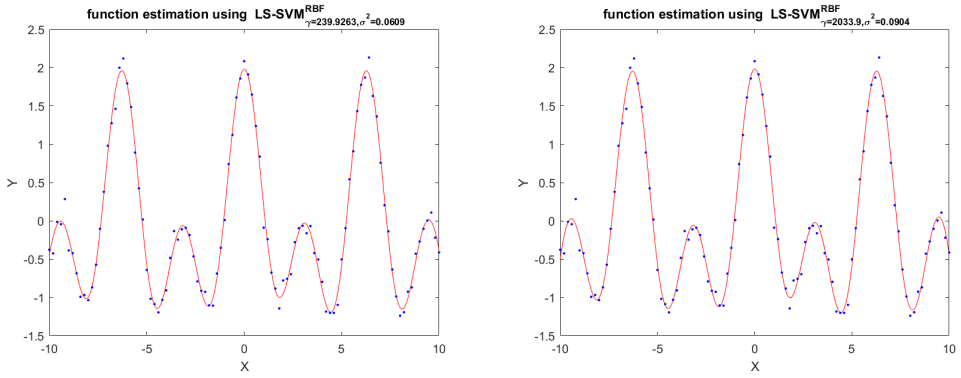
2.3 Hyper-parameter Tuning

The *tunelssvm* command was run 10 times with simplex optimization, and 10 times with gridsearch optimization. The results are very similar for both methods, as both found that a large γ and a σ^2 of around 0.06 leads to the best function approximation (Table 5). Overall, the σ^2 parameter stayed much more consistent than the γ parameter. Both methods found very similar values for σ^2, but differed slightly in their values for γ. Simplex optimization averaged a smaller γ over the 10 trial runs. Gridsearch optimization showed higher variance from run to run, especially for the γ parameter, but on average performed slightly better than simplex optimization as its average cost was lower. The gridsearch method works by defining a finite grid of combinations of γ and σ^2 values, and calculating the crossvalidation error for each combination of parameters. This is repeated with a more precise grid of parameter values each time a set number of times, or until a stopping criteria is met. The simplex method works by moving across the hypersurface of hyper-parameter values towards the minimum crossvalidation error in one of four defined movements (reflection/expansion/contraction/shrink). When a stopping criteria is met, the movements are stopped and the parameters where the minimum is found are returned. The gridsearch method is more computationally

intensive, while the simplex method is more susceptible to finding local minima.

Table 5: Results from 10 runs each of the *tunelssvm* command.

	Simplex			**Gridsearch**		
	gam	**sig2**	**cost**	**gam**	**sig2**	**cost**
minimum	190.8228	0.0344	0.0146	95.4125	0.0389	0.0141
maximum	1703.4	0.0841	0.0165	2033.9	0.0904	0.0167
average	430.4119	0.06149	0.0165	568.75583	0.06625	0.0153



(a) Simplex method (b) Gridsearch method

Figure 44: Function estimation on the training set for RBF kernel LSSVM with parameters optimized by two different methods, simplex optimization and gridsearch optimization.

2.4 Application of the Bayesian Framework

2.4.1 Regression

The Bayesian framework for LS-SVM computes posterior probability, which is the probability of the data given the model. It operates on a three-level principle, of which a schematic visualization is shown in the course notes. At level 1, the α and β parameters are found by maximizing the posterior with respect to these parameters. The evidence from this level equals the likelihood of level 2, where the γ parameter is found by maximizing the posterior with respect to γ. Then the evidence from level 2 is the likelihood of level 3, where the σ parameter is found by maximizing the posterior with respect to σ. This allows for the computation of error bars in regression models, as shown in Figure 45.

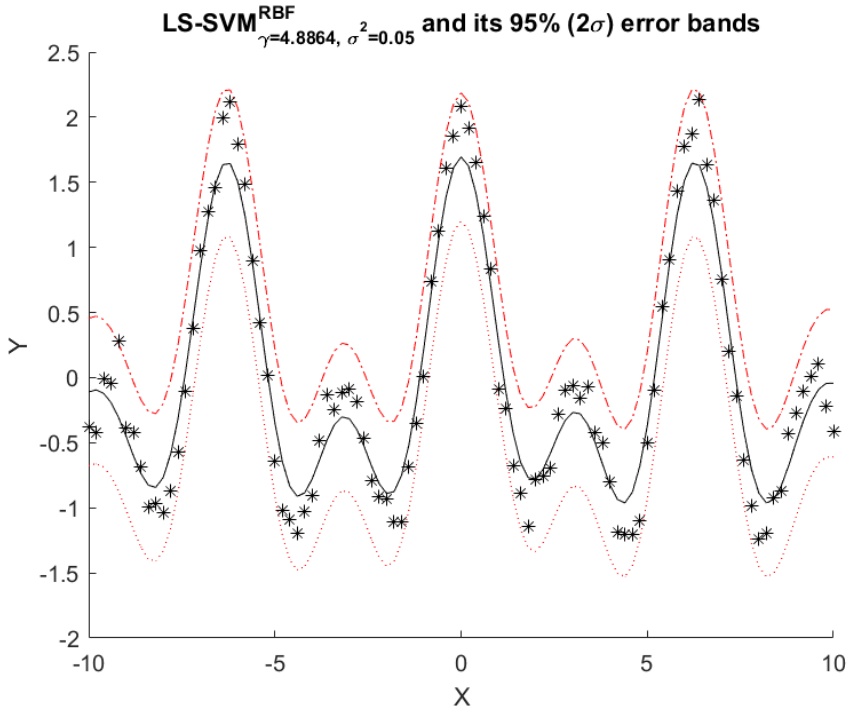


Figure 45: Performance of LS-SVM regression with a Bayesian framework on the training set.

2.4.2 Classification

Bayesian framework can also be applied to classification. Figure 45 shows the results of LS-SVM classification with a Bayesian framework on the training set of the Iris dataset. Pink corresponds to a high probability that this part of the input space belongs to the positive class, while blue corresponds to a low probability of belonging to the positive class. The more intense the coloring, the stronger (pink) or weaker (blue) the chance is that data points in this area would be classified to the positive class. A low probability of belonging to the positive class implies a high probability of belonging to the negative class. Different values for γ and σ^2 were investigated (Figure 47). In general, as γ increases, the uncertainty increases, as there is less area with intense blue or pink coloring. At high γ, most of the input space has probability of belonging to a certain class of around 50%. As σ^2 increases, the amount of plotted area that is blue increases, meaning the negative classification area increases.

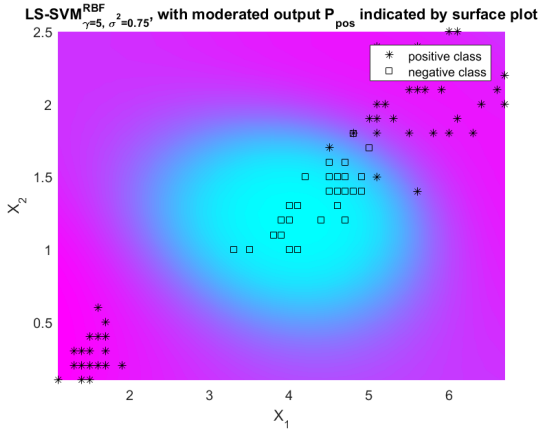
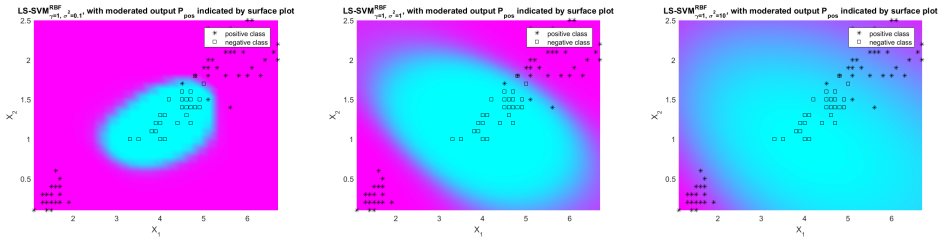
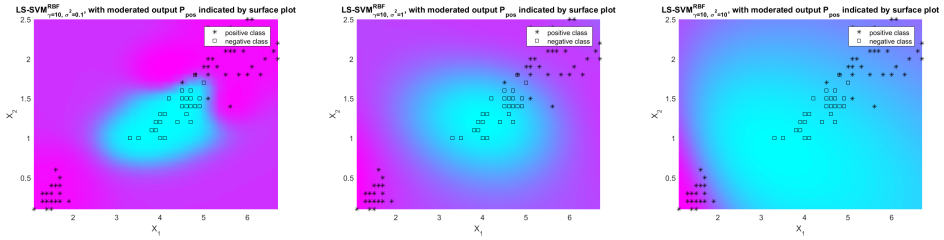


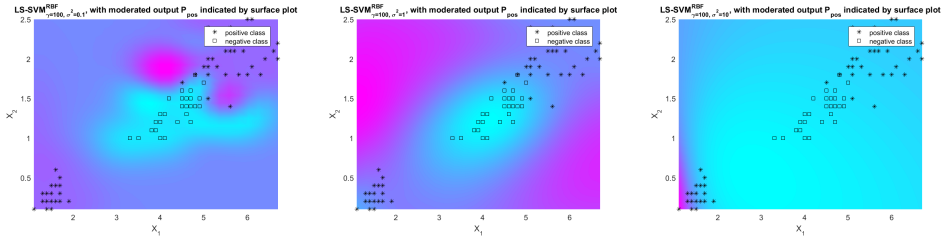
Figure 46: Performance of LS-SVM classification with a Bayesian framework on the Iris dataset.



(a) $\gamma = 1$, $\sigma^2 = 0.1$ (b) $\gamma = 1$, $\sigma^2 = 1$ (c) $\gamma = 1$, $\sigma^2 = 10$



(d) $\gamma = 10$, $\sigma^2 = 0.1$ (e) $\gamma = 10$, $\sigma^2 = 1$ (f) $\gamma = 10$, $\sigma^2 = 10$



(g) $\gamma = 100$, $\sigma^2 = 0.1$ (h) $\gamma = 100$, $\sigma^2 = 1$ (i) $\gamma = 100$, $\sigma^2 = 10$

Figure 47: Performance over different combinations of σ^2 and γ parameters for an LS-SVM classification with a Bayesian framework on the Iris dataset.

2.4.3 Automatic Relevance Determination

A Bayesian framework can also be used to select the most relevant inputs, using backward selection. A dataset with three variable was created, and then a response variable is generated using only the first variable. As expected, the Bayesian framework chooses only the first variable as a relevant input for response variable. Figure 48 shows the cost decreases from the full model with three variables to the minimum cost at the model with only X1 as a variable. Input selection can be done using the *crossvalidate* function by making models with all combinations of variables, and choosing the inputs from the model with the lowest test MSE.

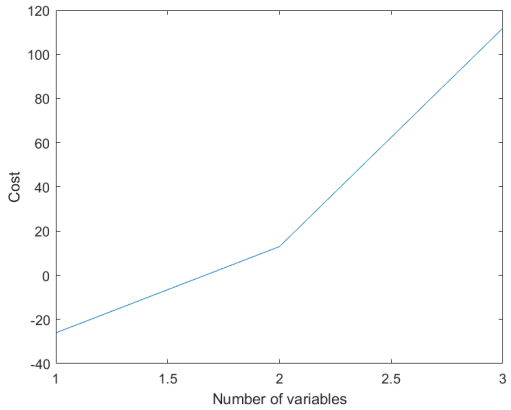


Figure 48: Performance of LS-SVM classification with a Bayesian framework on the Iris dataset.

2.5 Robust Regression

The effect of outliers on LS-SVM regression in Figure 49. Outliers cause the first peak to be slightly lower than the other two peaks, and cause a poor regression fit in the area where X = -2.

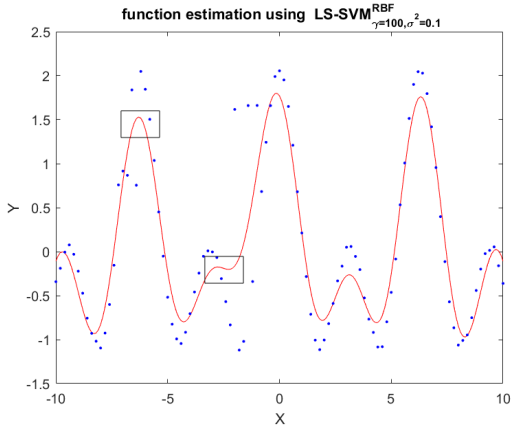
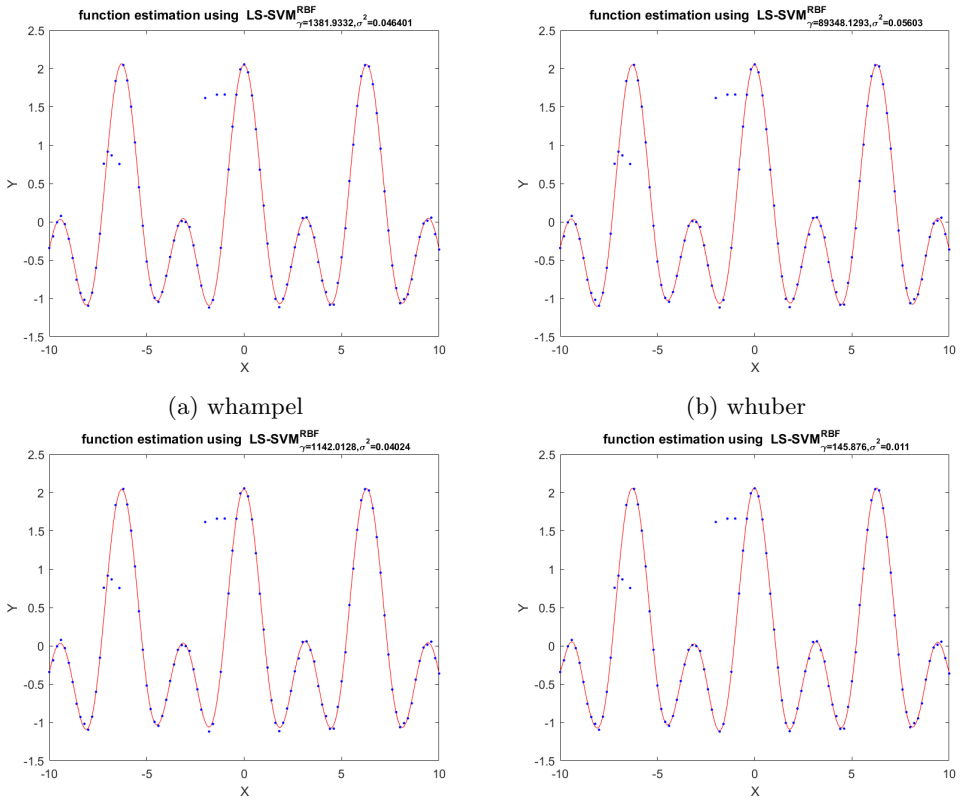


Figure 49: LS-SVM regression model on a dataset with outliers. The areas outlined in black circles are where the regression line deviates from the regression line trained on the same data without outliers.

For better performance in the presence of outliers, robust regression was performed using the *robustlssvm* function. This generated weighted LS-SVM regression models using different weighting schemes. All four weighting schemes produce very similar function estimations, and are clearly better at fitting the true function in the presence of outliers than regular LS-SVM regression (Figure 50). Mean absolute error is preferred over mean squared error because by not squaring the error, the mean absolute error is less influenced by large errors. Therefore, in datasets that contain a few points with large error (outliers), mean absolute error is a better choice.



(a) whampel (b) whuber

(c) wlogistic (d) wmyriad

Figure 50: Robust LS-SVM regression models generated by different weighting functions.

2.6 Homework Problems

2.6.1 Santa Fe Laser Dataset

Time series prediction was applied to the Santa Fe laser time series data. The data was split into a 1000 data point training set and a 200 data point test set plotted in Figure 51.

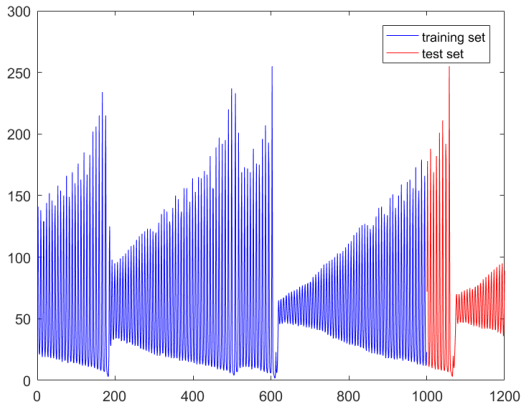


Figure 51: Training set and test set of the Santa Fe laser time series data.

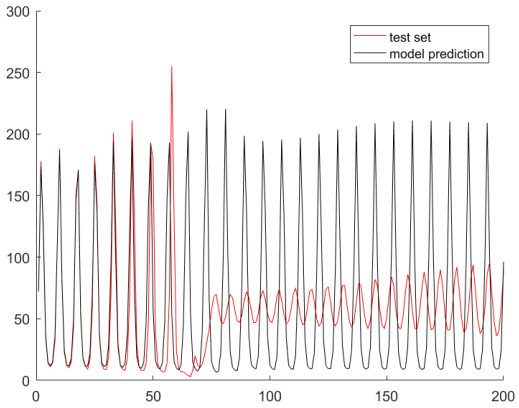


Figure 52: Prediction on test set for model with parameters given in the exercise.

First, I ran the code given in the exercise (with nb changed to 200 to predict the full test set). The *order*, γ and σ^2 parameters were set to 10. The performance of the model on the test set is shown in Figure 52. Clearly, the prediction is not very good and there is room for improvement. To find the best model, models were generated with order parameters from 10 to 300 in steps of 10. The γ and σ^2 parameters were optimized for each model using the *tunelssvm* command with 10 fold crossvalidation on the training set. Each LS-SVM was trained on the training set using the optimized hyperparameters, then used to predict the next 200 time points. Model performance was evaluated by the mean squared error between these 200 predictions and the test set. The minimum mean squared error was found when the order of the model was 110 (Figure 53).

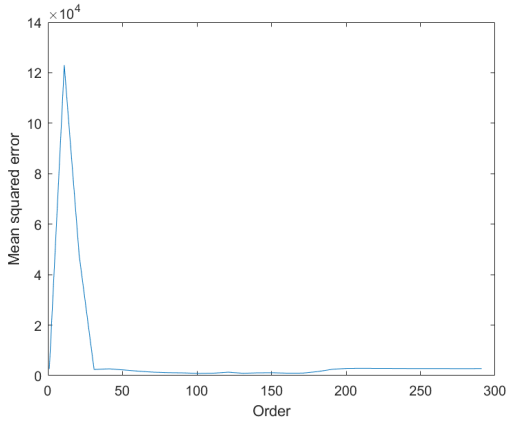


Figure 53: Means squared error on the test set for models according to their order parameter.

The performance on the test set of the model with an order of 110 and optimized hyperparameters was compared to the performance of the model with an order 50 and optimized hyperparameters (Figure 54, 55). The order 110 model is more flexible and does a better job of modeling the large decrease in intensity that

occurs just after the 50th time point in the test set.

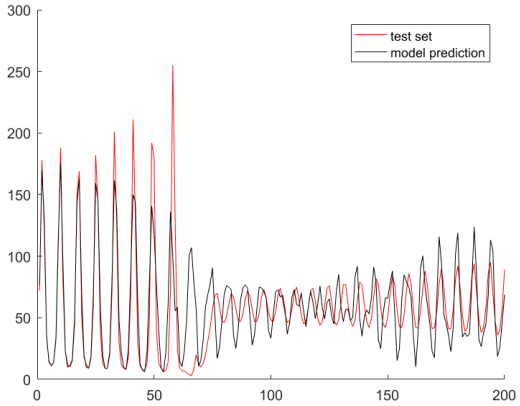


Figure 54: Order = 110

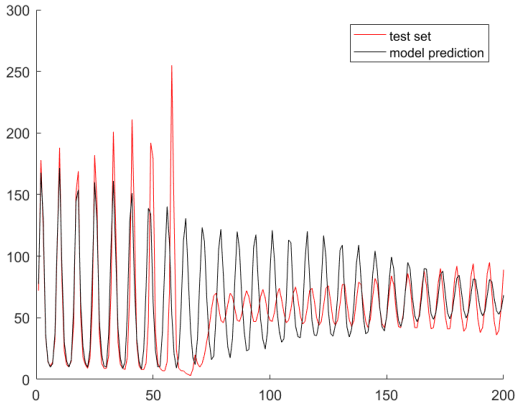


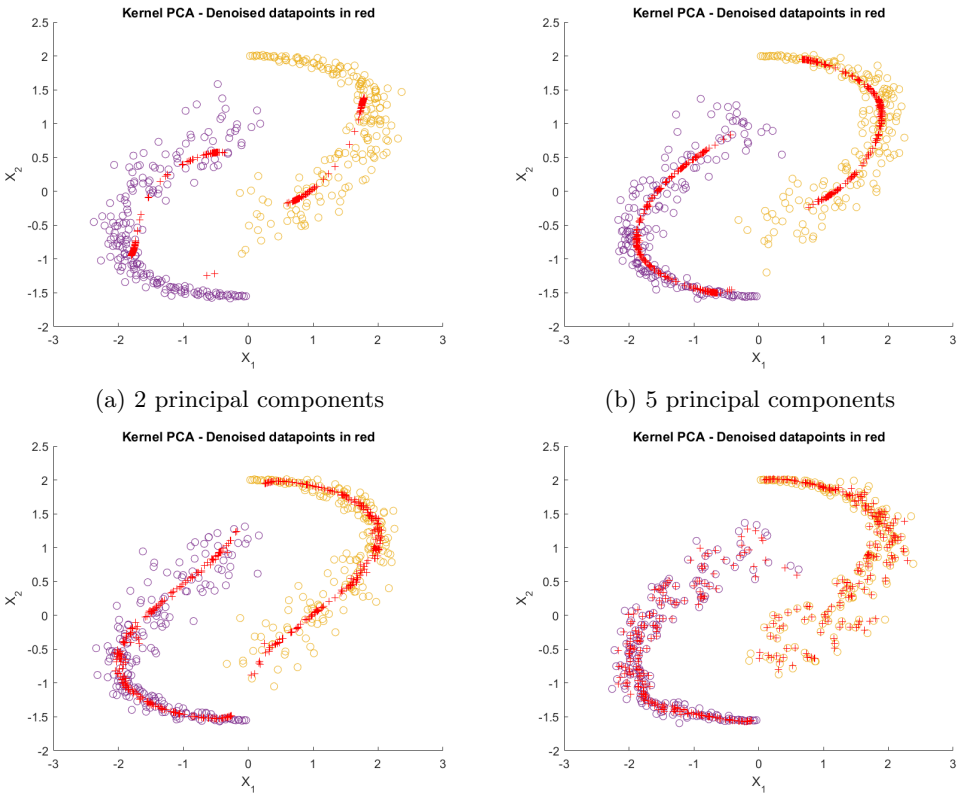
Figure 55: Order = 50

It would be sensible to use the performance of this recurrent prediction on the validation set to optimize hyperparameters and the model order if the validation set is representative of the overall time series data. If the validation set only contains time points that are not at all representative of future time points, then the model will perform poorly on future predictions.

3 Exercise Session 3: Unsupervised Learning

3.1 Kernel Principal Component Analysis

The *kpca script* was run and a toy structured dataset was generated of 400 points with a dispersion of 0.3. The goal of kpca in this section is to reconstruct the data in a way that eliminates the noise. A high number of PC's (PC = 20) leaves significant noise still left in the data (Figure 56). A low number of PC's causes the denoised data points to form small clusters within the overall cluster of the true data points. The best choice for denoising the data appears to be somewhere between 5 and 10 principal components. A difference between linear PCA and kernel PCA is the maximum number of possible principal components. The maximum number of PCs in a linear PCA equals dimension of the input space, while a kernel PCA allows for a higher number of PCs than inputs. The results of a linear PCA are shown in Figure 57. The kernel PCA is obviously better at finding the structure in this dataset. A crossvalidation approach with the data split into a training set and test set could be used to tune the number of components and the kernel hyper-parameter. This approach would seek to find the component number and hyperparameter that minimizes the mean squared error between the denoised datapoints and the datapoints in the test set.



(a) 2 principal components

(b) 5 principal components

(c) 10 principal components

(d) 20 principal components

Figure 56: Kernel PCA (σ^2=0.4) with different numbers of principal components. The denoised data points are in red.

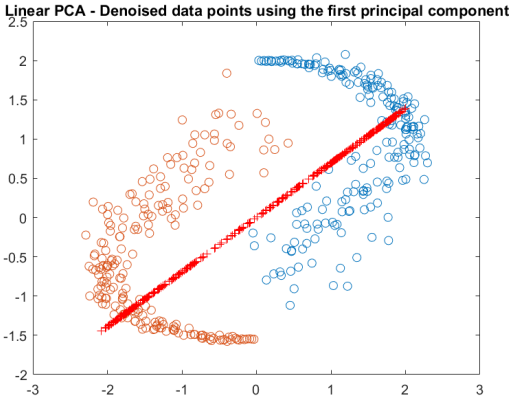
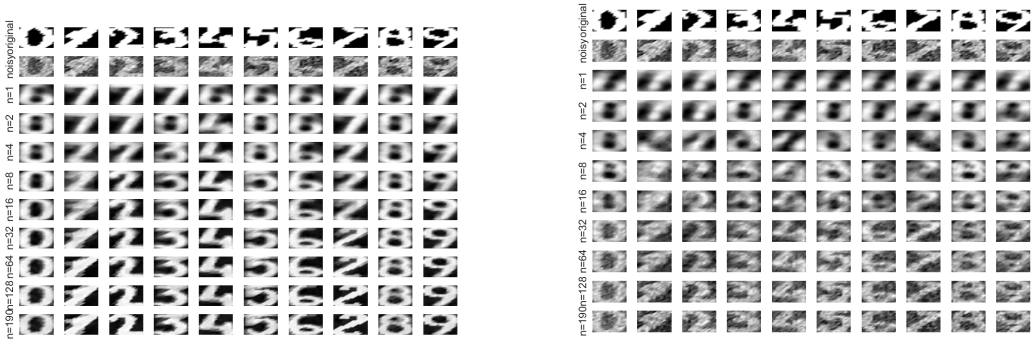


Figure 57: Linear PCA with denoised data points in red.

3.2 Handwritten Digit Denoising

The results of using a kernel PCA and a linear PCA with different number of principal components to denoise a set of handwritten digits are shown in Figure 58. Overall, the kernel PCA is more successful at removing the noise from the data. At a high number of PCs increases, the kernel PCA returns an image that most closely resembles the original, noiseless image, while the linear PCA gives the noisy image. When the number of PCs is low, both the kernel and linear PCAs seem to be biased towards 6's and 7's. When the number of PCs increases to around 16, all the digits are distinguishable from each other (much more dramatically so for the RBF kernel compared to the linear kernel). When the number of PCs is very high, all the digits are easily distinguishable, except for the 7 which starts looking like a 2 for the kernel PCA.



(a) Kernel PCA

(b) Linear PCA

Figure 58: Denoising handwritten digits with added noise using kernel PCA and linear PCA with different numbers of principal components.

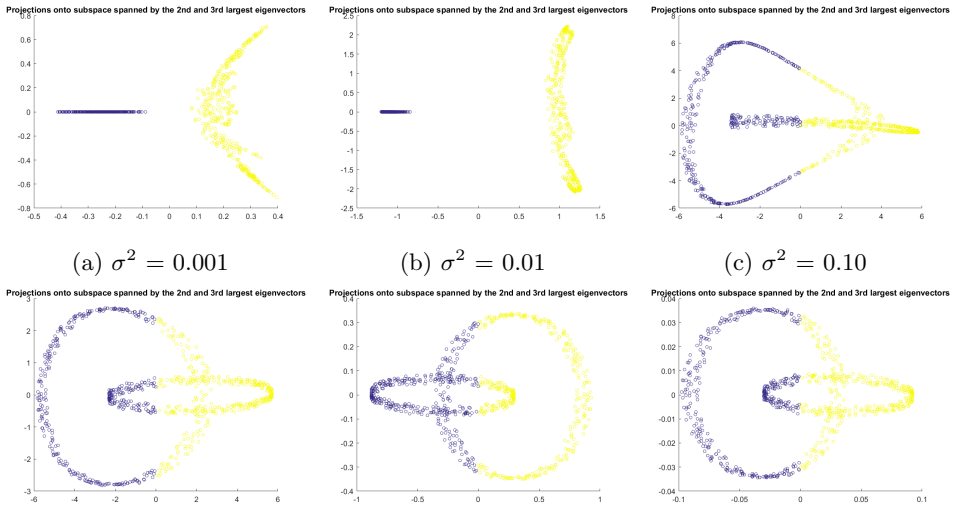
3.3 Spectral Clustering

The difference between clustering and classification is that clustering is unsupervised learning while classification is supervised. With clustering, the labels for the data points are unknown. The goal is to divide the dataset into clusters of similar data points, usually based on a distance measure of similarity. With classification, labels are known and the goal is to assign one of these labels to new data points. Different values of the σ^2 hyperparameter result in different clustering of the data points that make up the two rings. Ideally, all data points in one ring would be assigned to one cluster, while all the data points in the other ring would be assigned to another cluster. This perfect clustering occurs when σ^2 is from 0.001 to 0.01 (Figure 59). As σ^2 gets larger, the data points from one ring that are inside the other ring are assigned to the wrong cluster. This influence of the σ^2 hyperparameter is also seen in Figure 60, which shows the clustering as projections onto the subspace spanned by the 2nd and 3rd largest eigenvectors. The projections show that 0.01 is the best choice for the σ^2 hyperparameter because it has the tightest clustering.

(a) $\sigma^2 = 0.001$ (b) $\sigma^2 = 0.01$ (c) $\sigma^2 = 0.10$

(d) $\sigma^2 = 1.0$ (e) $\sigma^2 = 10$ (f) $\sigma^2 = 100$

Figure 59: Performance over different combinations of σ^2 and γ parameters for an LS-SVM classification with a Bayesian framework on the Iris dataset.



(a) $\sigma^2 = 0.001$ (b) $\sigma^2 = 0.01$ (c) $\sigma^2 = 0.10$

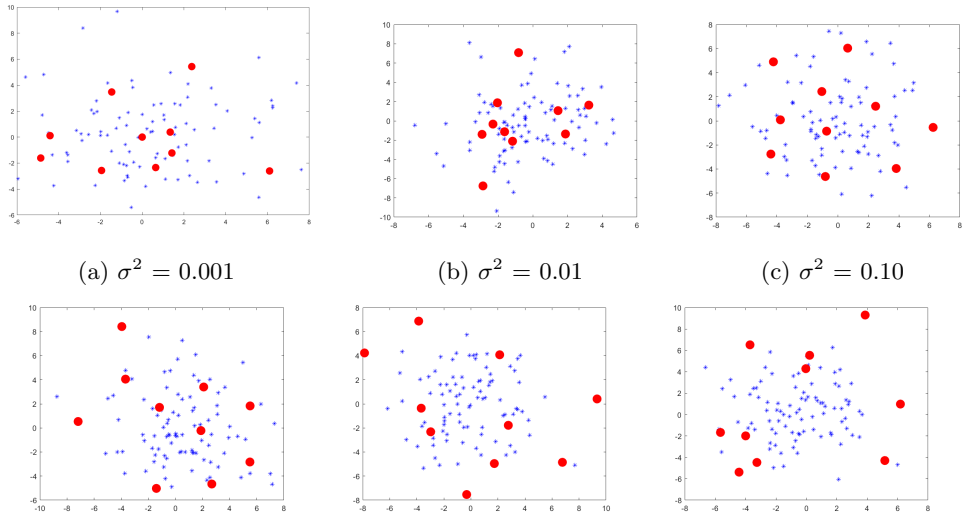
(d) $\sigma^2 = 1.0$ (e) $\sigma^2 = 10$ (f) $\sigma^2 = 100$

Figure 60: Performance over different combinations of σ^2 and γ parameters for an LS-SVM classification with a Bayesian framework on the Iris dataset.

3.4 Fixed-size LS-SVM

1. The *fixedsize_script1* was run, which uses a Nystrom approximation to approximate the feature space with a fixed subset of data points. A range of values for the σ^2 parameter for the approximation were tested to examine the influence of this parameter on the location of the subset of data points. (Figure 61). As

σ^2 increased, the approximation data points moved out towards the perimeter of the data cloud. The data points of the subset seem to be most evenly spread out when σ^2 was from 0.1 to 1.0. At a lower σ^2, the subset data points seem to be more concentrated in the center of the cloud of data points.



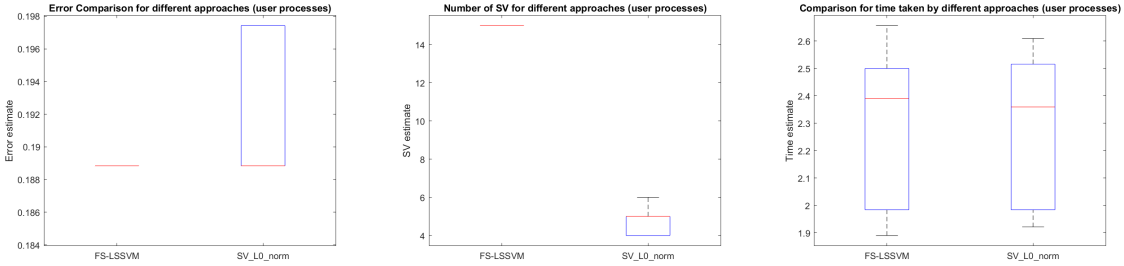
(a) $\sigma^2 = 0.001$ (b) $\sigma^2 = 0.01$ (c) $\sigma^2 = 0.10$

(d) $\sigma^2 = 1.0$ (e) $\sigma^2 = 10$ (f) $\sigma^2 = 100$

Figure 61: Approximation of the feature space through subset selection by Nystrom approximation with different σ^2 parameters. The subset is represented by the 10 red data points.

2. The algorithm converges when a subset is found that best represents the original data in the feature space. This means that a model built in the feature space from the subset will best approximate the model built from all original data in the original space. The choice of data points that make up the subset is done through entropy measurements.

3. When a sparser solution is preferred, an l_0 penalty is applied to a fixed-size LS-SVM. The l_0 approximation has a lower number of support vectors, the same error, and takes about the same amount of time.



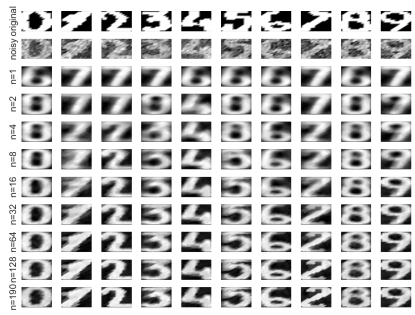
(a) Error comparison (b) Support vectors (c) Time

Figure 62: Comparison of the results of a fixed-size LS-SVM to an l_0 approximation.

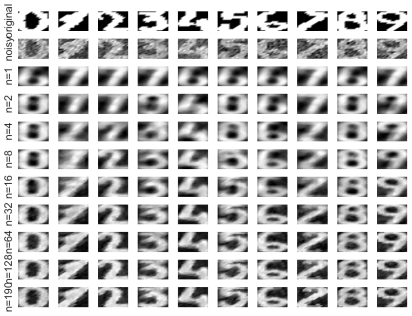
3.5 Homework Problems

3.5.1 Handwritten Digit Denoising

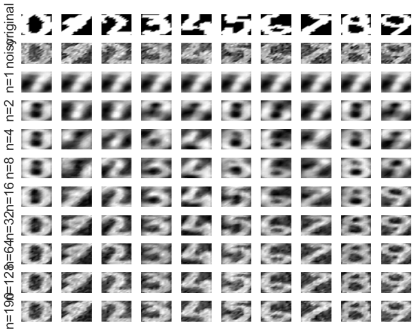
Using the rule of thumb and a *sigmafactor* of 0.7 (the default in the *digitsdn* script), the σ^2 hyperparameter is set to around 35. I tested the impact of a much larger σ^2 by setting the hyperparameter to 50, 100, 500, and 1000 (Figure 63). The denoising ability of the RBF kernel PCA got worse as σ^2 increased, and at a very high σ^2 the RBF kernel PCA approximates the linear PCA from Figure 58.



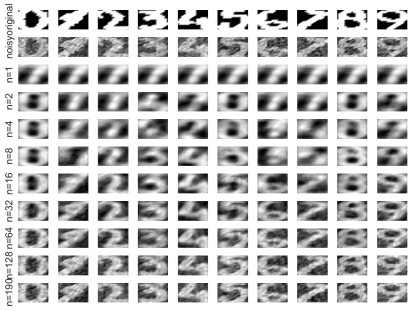
(a) $\sigma^2 = 50$



(b) $\sigma^2 = 100$



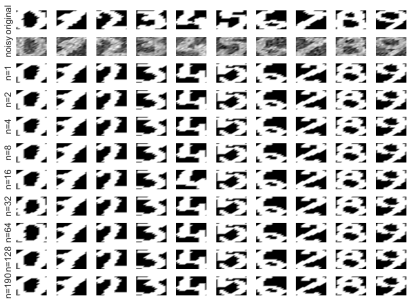
(c) $\sigma^2 = 500$



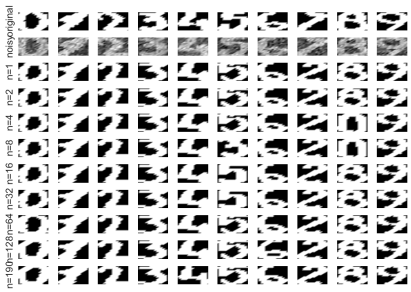
(d) $\sigma^2 = 1000$

Figure 63: Digit denoising with a kernel PCA with different values for σ^2.

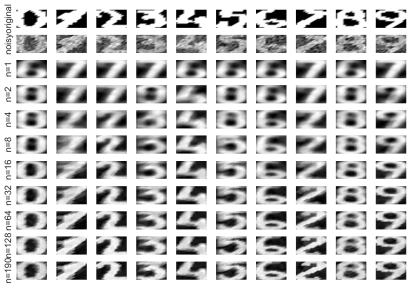
Next the σ-factor parameter was changed to equispaced values in logarithmic scale from -2 to 2. Since the σ-factor is just a scalar for the σ^2 parameter, changing the σ-factor gives very similar results to the previous question (Figure 64). A very low σ-factor results in a more flexible kernel that gives good results, except for the 7 digit. A very high σ-factor results in a less flexible kernel that gives very poor results, as the σ-factor = 100 figure shows that for some digits the original data was not recovered. The best choice for the σ-factor parameter appears to be between 0.1 and 1.



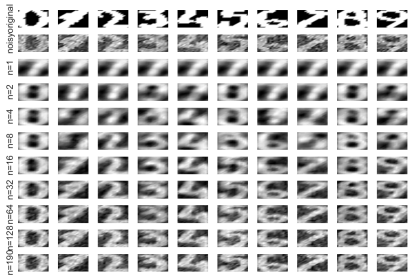
(a) σ-factor = 0.01



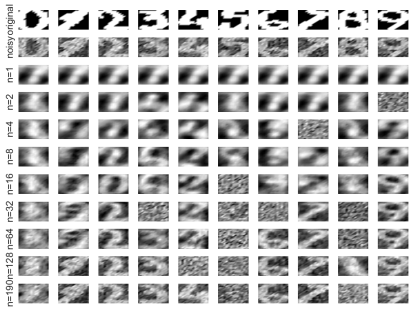
(b) σ-factor = 0.1



(c) σ-factor = 1



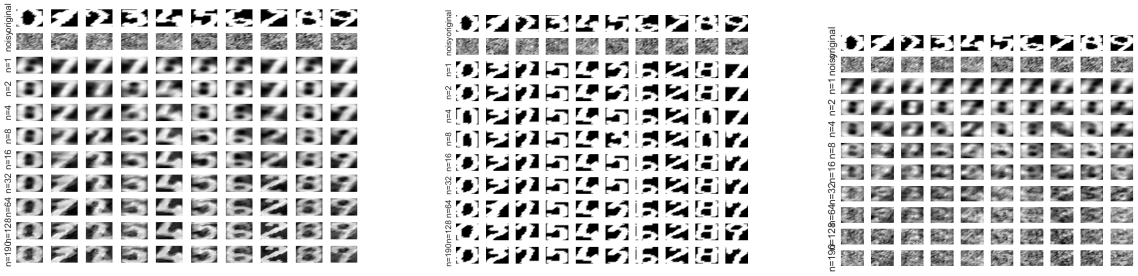
(d) σ-factor = 10



(e) σ-factor = 100

Figure 64: Digit denoising with a kernel PCA with different values for σ-factory parameter.

Then noise was added to the data by increasing the *noisefactor* to 1. The linear kernel really struggled with the added noise, as it was not able to reconstruct the original (de-noised) images (Figure 65). Two different σ-factor values were tested, with the σ^2 parameter kept as the default. A σ-factor of 0.1 seemed to result in overfitting, while a σ-factor of 1 performed well in denoising the data.



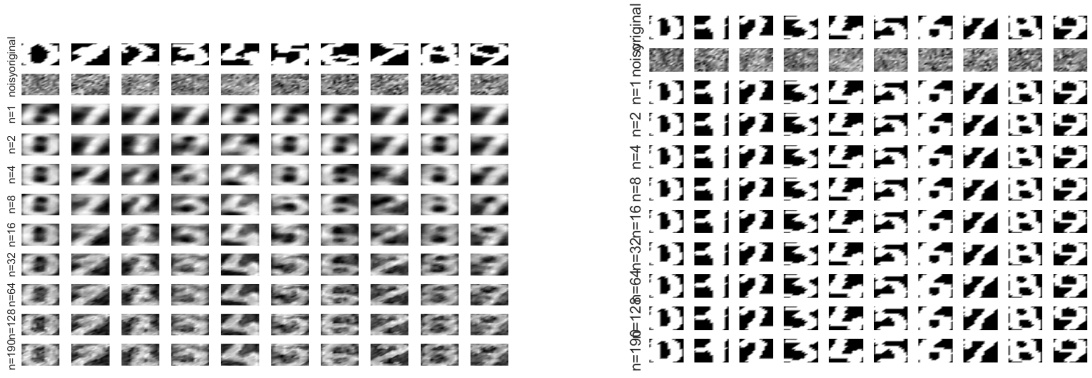
(a) σ-factor = 1 (b) σ-factor = 0.1 (c) Linear

Figure 65: Two examples of example of digit denoising for an RBF kernel compared to a linear kernel for a *noisefactor* of 1.0.

Finally, the MSE on the validation sets Xtest1 and Xtest2 were calculated for different values of σ^2 (Table 6). Overall, performance was better on the Xtest2 set, where the minimum error occurred when σ^2 was equal to 1. The performance on Xtest1 got better as σ^2 increased, as the minimum error on this validation set was when σ^2 equaled 100. The two best performing models are shown in Figure 66 , where we can clearly see that the reconstruction of Xtest2 was better.

Table 6: MSE on different validation sets for different σ^2 values.

	Sigma				
	0.01	**0.1**	**1**	**10**	**100**
Xtest1	0.5258	0.5111	0.2087	0.2053	0.1895
Xtest2	0.5569	0.5155	2.62E-27	1.07E-05	0.0684

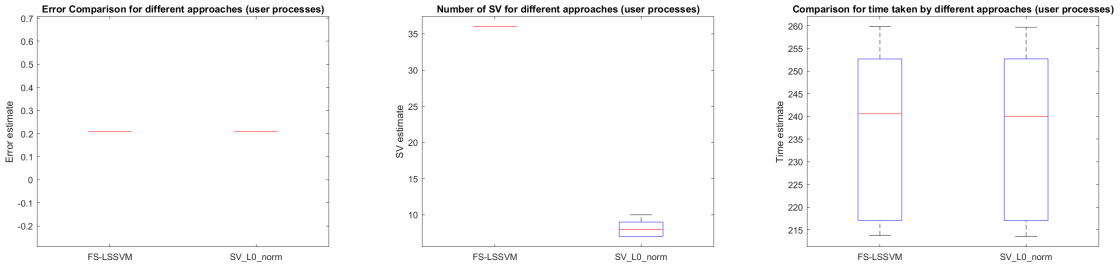


(a) Xtest1: $\sigma^2 = 100$ (b) Xtest2: $\sigma^2 = 1$

Figure 66: Digit denoising on two different validation sets.

3.5.2 Shuttle (statlog)

The *fslssvm_script* was adjusted and used for classification on the Shuttle dataset. The Shuttle data was split into a training set (70 percent) and a test set (30 percent). A linear fixed-size LS-SVM model was generated with and without l_0 approximation. The fixed-size LS-SVM with l_0 approximation had a much lower number of support vectors, but higher variance in the number of support vectors chosen. The time and error between both models was very similar.

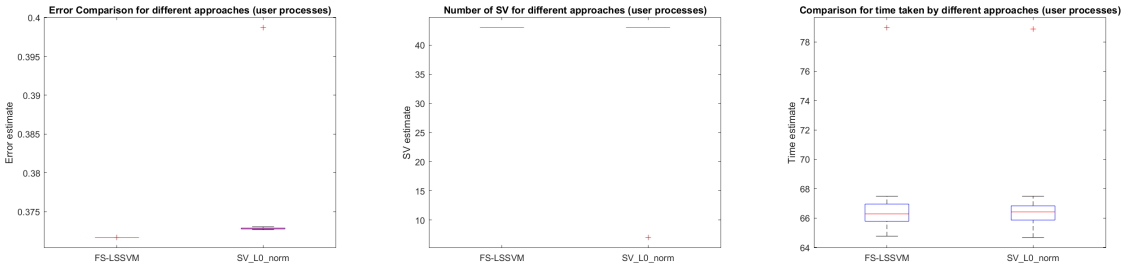


(a) Error comparison (b) Support vectors (c) Time

Figure 67: Comparison of the results of a fixed-size LS-SVM to an l_0 approximation on the Shuttle dataset.

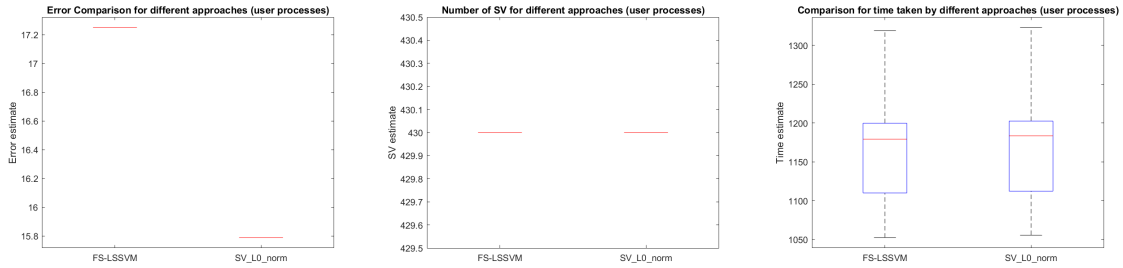
3.5.3 California

The *fslssvm_script* was adjusted and used for regression on the California dataset. The California data was split into a training set (70 percent) and a test set (30 percent). Fixed-size LS-SVM models with linear and polynomial kernels were generated with and without l_0 approximation. The more complex polynomial models had more support vectors and the run time was a lot longer. Interestingly, there was no difference in number of support vectors for the models with l_0 approximation compared to those without for both the linear and polynomial kernel. For the linear kernel, the FS-LSSVM with no penalty had a lower error, while for the polynomial kernel, the FS-LSSVM with the penalty applied has a lower error.



(a) Error comparison (b) Support vectors (c) Time

Figure 68: Linear kernel: comparison of the results of a fixed-size LS-SVM to an l_0 approximation on the California dataset.



(a) Error comparison (b) Support vectors (c) Time

Figure 69: Polynomial kernel: comparison of the results of a fixed-size LS-SVM to an l_0 approximation on the California dataset.