Support Vector Machines: Methods and Applications

Dimitrios Chatzakis r0977164

August 2024

First Exercise Session

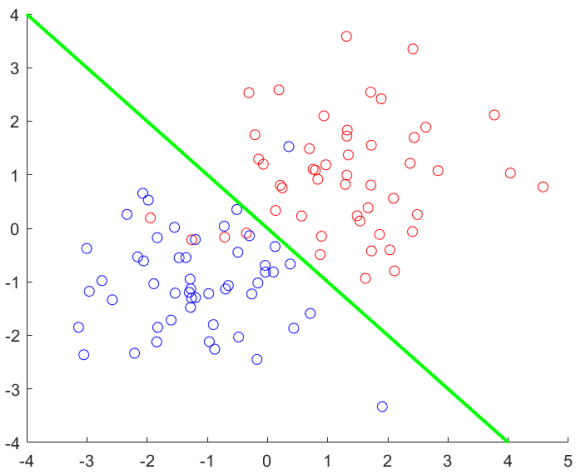

Figure 1: Optimal Separation of Gaussian data.

Q1.1: Obtain a line to classify the data by using what you know about the distributions of the data. In which sense is it optimal?

Since the data are Gaussian distributions centered at (1,1) and (-1, -1), the simplest line that would split the dataset optimally would be the y=-x line (Figure 1).

Q1.2.1: What do you observe when you add more data points to the dataset- both on the right and on the wrong side of the hyperplane. How does it affect the classification hyperplane?

One would expect the added number of points to push the linear boundaries of the SVM closer to the true y=-x line. The resulting figures 2 and 3 are presented below. As we can see the slope has increased, tending more closely to the optimal separator. The same difference using the RBF kernel can be seen in Figures 4 and 5 below.

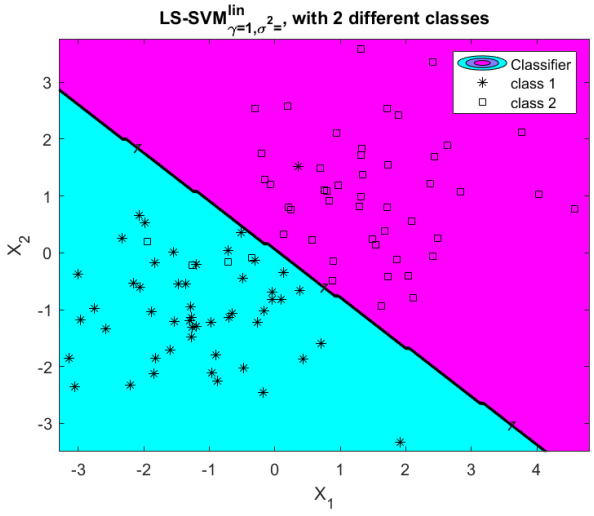

Figure 2: Linear SVM trained on old data.

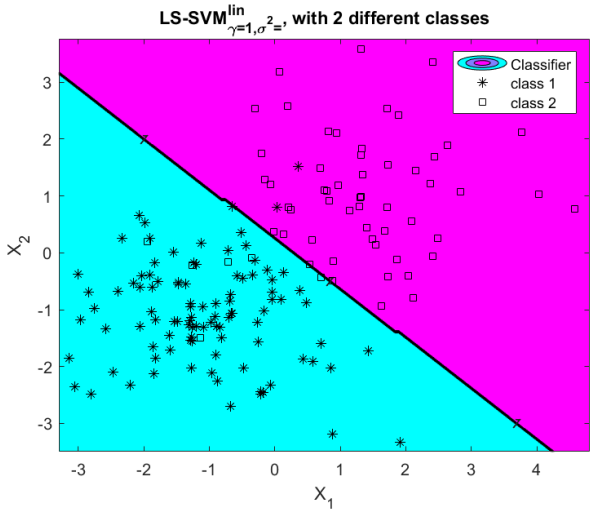

Figure 3: Linear SVM with additional data.

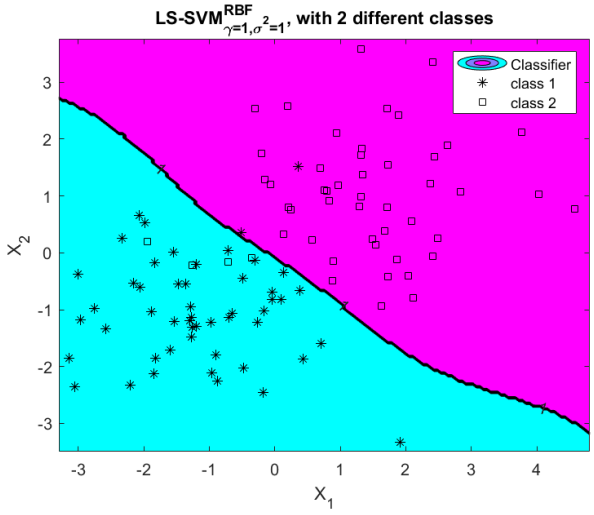

Figure 4: RBF SVM trained on old data.

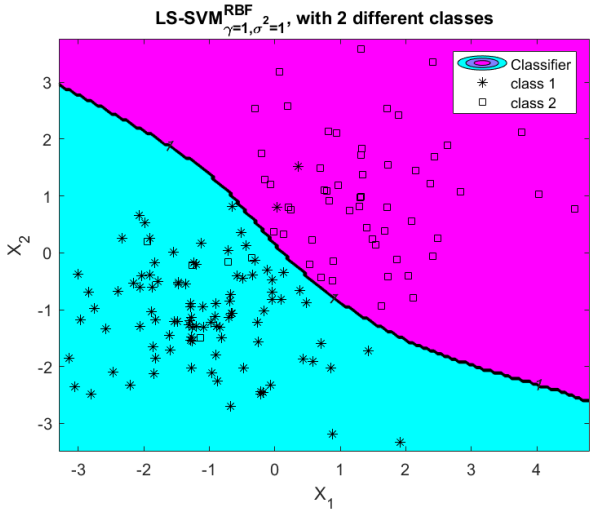

Figure 5: RBF SVM with additional data.

In the case of the RBF kernel the increase in data makes the resulting line straighter decreasing the changes in curvature from 4 to 2 which is to be expected since the extra data make the approximation of the optimal distribution more robust.

Q1.2.2: Try out different values of the regularization hyperparameter C and the kernel parameter sigma. What is the role of the parameters? How do these parameters affect the classification outcome?

.

The equation that the SVM(right) or LS-SVM(left) is trying to minimize is:

$$\min_{w,b,e} \mathcal{J}(w,e) = \frac{1}{2}w^T w + \gamma \sum_{k=1}^{N} e_k^2 \qquad (1)$$

$$y_k[w^T \varphi(x_k) + b] = 1 - e_k, \qquad (2)$$

$$k = 1, \ldots, N \qquad (3)$$

$$\min_{w,b,\xi} \quad \frac{1}{2}w^T w + C \sum_{k=1}^{N} \xi_k \qquad (4)$$

$$y_k(w^T x_k + b) \geq 1 - \xi_k, \quad k = 1, \ldots, N \qquad (5)$$

$$\xi_k \geq 0, \quad k = 1, \ldots, N \qquad (6)$$

The first term allows for the maximization of the margins while the second represents the penalty for miss-classification. In the SVM it is the L1 norm while in the case of LS-SVM is the L2 and that is why it is called Least Squares SVM. A high γ value would put more emphasis on the second term, making the model classify the training data correctly. This usually leads to overfitting, as the training data is classified correctly, but the generalization of the model is very weak. On the other hand a low γ would lead to a model that prioritizes more maximizing the margins, leading to better generalization but could end up underfitting. In the figures below I display the results for adding 10 and and 49 data to my inputs for the two classes, making them 60 and 100 in size respectively (Figures 6-8).

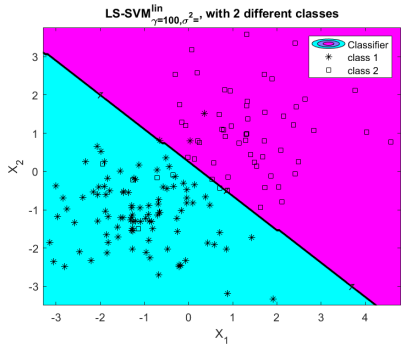

Figure 6: γ=100, linear kernel

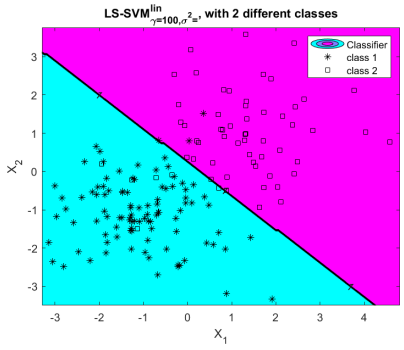

Figure 7: γ=1, linear kernel

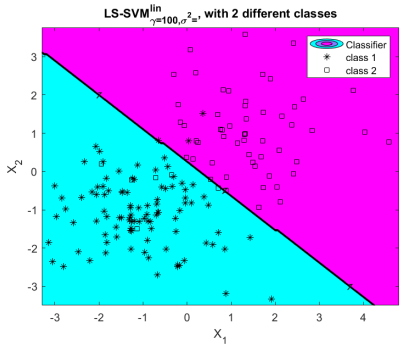

Figure 8: γ=0.01, linear kernel

As it can be seen the difference in the linear kernel is not big, as can be expected since the linearity of the model accommodates well the data. In the case of changing the γ parameter in the RBF kernel setting (Figures 9-11), it can be discerned that the model overfits the training data as the γ decreases to the point that it mainly predicts the majority class since the cost of miss-classification is very low:

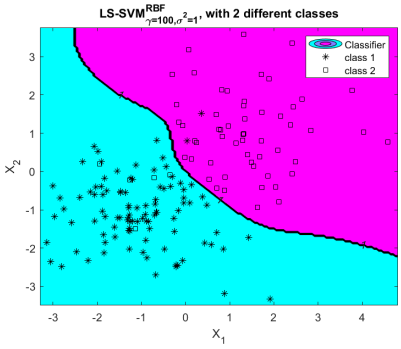

Figure 9: γ=100, RBF kernel

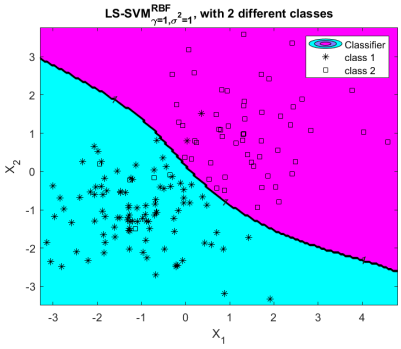

Figure 10: γ=1, RBF kernel

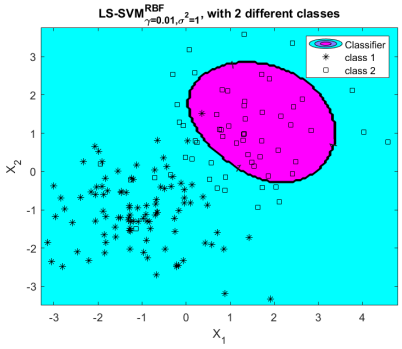

Figure 11: γ=0.01, RBF kernel

After changing the σ^2 parameter in a similar fashion for the RBF we obtain Figures 12-14.

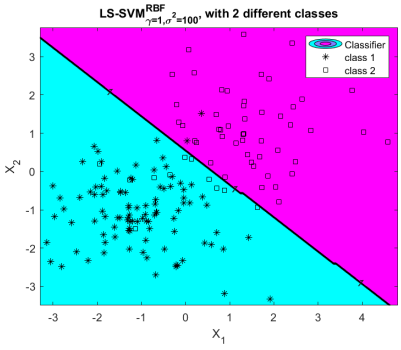

Figure 12: σ^2=100, RBF kernel

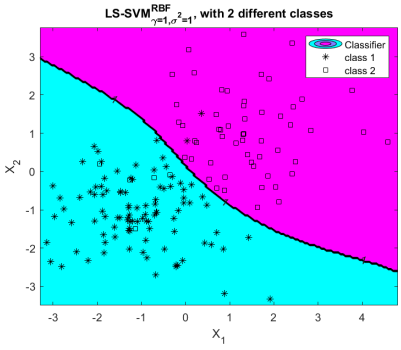

Figure 13: σ^2=1, RBF kernel

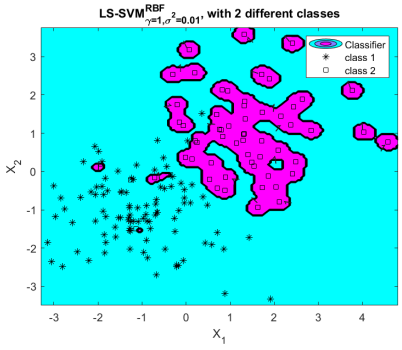

Figure 14: σ^2=0.01, RBF kernel

Since the σ^2 parameter determines the variance of each data point we can see that decreasing it makes the model more non-linear to the point that in the lower case the variance is so low in the minority class that islets form, clearly overfitting the data. Higher sigmas, spread the effective radius of the kernel, making the non-linear decision boundary smoother.

Q1.2.3: Compare classification using the linear kernel with classification using the RBF kernel. Which performs better? Why?

In the case of the training data the non-linearity of the RBF makes it more adaptable and better performing, since the data is limited and thus not fully representative of the underlying distributions with accuracies of 95% to 95.63% for the linear and RBF kernel respectively.

After creating a balanced test dataset of 100 points, contrary to common sense the RBF kernel achieves better results with 95% to 97% test set accuracy, which is something within statistical margins. Increasing the points to 1000, makes the linear model perform better with 92.2% to 92.1% accuracy respectively.

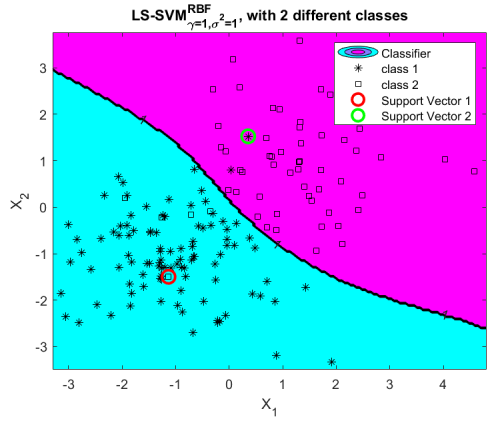

Figure 15: Optimal Separation of Gaussian data.

Q1.2.4: What is a support vector? When does a particular datapoint become a support vector? When does the importance of the support vector change? Illustrate visually. Note that a support vector is indicated by a large circle with bold-lined circumference and that its importance is proportional to the size in the online application.

A support vector is a datapoint that lies the closest to the decision boundary. These are the most important datapoints in the SVM method since they decide the classification boundary of the method. In the case of the typical SVM they would lie on the decision boundary, however when accounting for the miss-classification errors they could lie inside the margin area or be even miss-classified as is the case in our data (Figure 15). To state it also more mathematically, they are the points for which the alphas in the dual form are non zero, so they don't introduce sparseness.

Q1.2.5: What is the role of parameters C and sigma? What happens to the classification boundary if you change these parameters. Illustrate visually.

See answer to Q1.2.2. C (or γ) is the regularization parameter and sigma the variance that is assumed for the RBF kernel:

$$K(x, x') = \exp\left(-\frac{\|x - x'\|^2}{\sigma^2}\right) \tag{7}$$

Q1.2.6: What happens to the classification boundary when sigma is taken very large? Why?

See answer to Q1.2.2. The classification boundary becomes more linear since it assumes higher variance and the effective radius of each datapoint increases.

Q1.3.1.1: For the iris dataset try out a polynomial kernel with degree = 1,2,3,... and t = 1 (fixed gam = 1). Assess the performance on the test set. What happens when you change the degree of the polynomial kernel?

Degrees	1	2	3	6	10	15	20
Train Acc	58	94	96	96	97	99	99
Test Acc	45	95	100	100	100	95	80

Table 1: Table of performance of polynomial kernel for various degrees

The degrees I tried were 1, 2, 3, 6, 10, 15, 20. The performance (accuracy) on the train and test set was as in Table 1 to the left. As it can be discerned, the training performance increases, as it should given the better flexibility the model has to fit the data with higher degrees, since it can adjust its curvature more times. The test set however reaches a peak after which it begins to fall since the model overfits to the training data. Below in Figures 16-18 some characteristic examples are displayed.

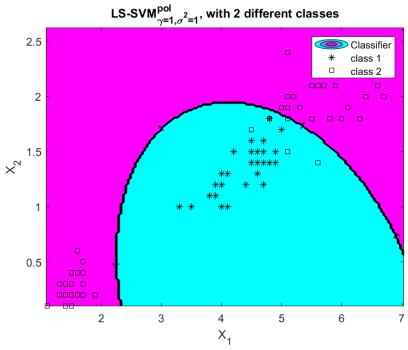

Figure 16: degree=2, poly kernel

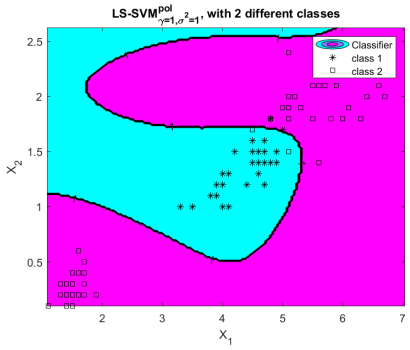

Figure 17: degree=6, poly kernel

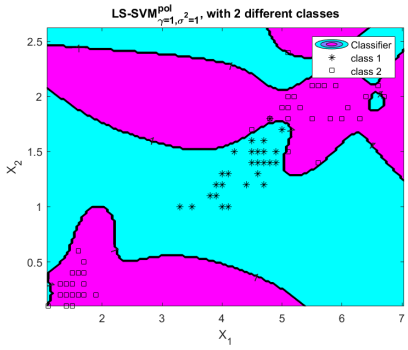

Figure 18: degree=15, poly kernel

Q1.3.1.2: Try out a good range of different sig2 values as RBF kernel parameters (fix gam = 1). Assess the performance on the test set. What is a good range for sig2?

σ^2	1000	100	10	1	0.1	0.01	0.001
Train Acc	67	67	95	95	96	99	99
Test Acc	50	50	100	100	100	90	70

Table 2: Table of performance of RBF kernel for various σ^2, keeping C=1.

The sig2 I tried were 1000, 100, 10, 1, 0.1, 0.01, 0.001. The performance (accuracy) on the train and test set was as in Table 2 to the left. As it can be discerned, the training performance increases, as it should given the better flexibility the model has to fit the data with lower sig2, since each training points' radius contains only itself. The test set however reaches a peak after which it begins to fall since the model overfits to the training data.Below in Figures 19-21 some characteristic examples are displayed. In Figure 19 the high σ makes all the points be classified as cyan, and is left uncolored. It can be seen that a range between 0.1 and 10 is safe for optimal results.

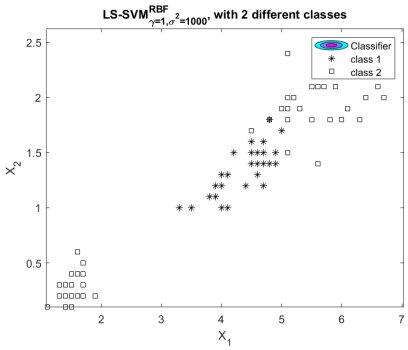

Figure 19: σ^2=1000, RBF kernel

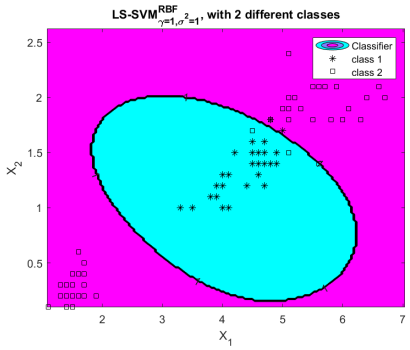

Figure 20: σ^2=1, RBF kernel

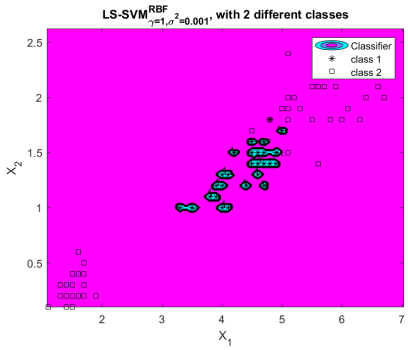

Figure 21: σ^2=0.001, RBF kernel

Q1.3.1.3: Fix a reasonable choice for the sig2 parameter and compare the performance using a range of gam. What is a good range for gam?

C	1000	100	10	1	0.1	0.01	0.001
Train Acc	99	99	98	96	96	67	67
Test Acc	95	95	100	100	90	50	50

Table 3: Table of performance of RBF kernel for various C values, keeping σ^2=0.1

The gammas I tried were 1000,100,10,1,0.1,0.01,0.001. The performance (accuracy) on the train and test set was as in Table 2 to the left. As it can be discerned, the training performance increases, as it should given the better flexibility the model has when the C parameter is high, penalizing more the miss-classification error. The test set again reaches a peak after which it begins to fall since the model overfits to the training data. Below in Figures 22-24 some characteristic examples are displayed.

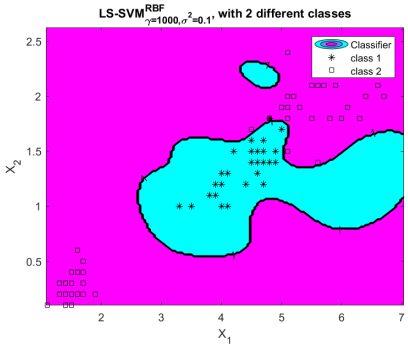

Figure 22: C=1000, RBF kernel

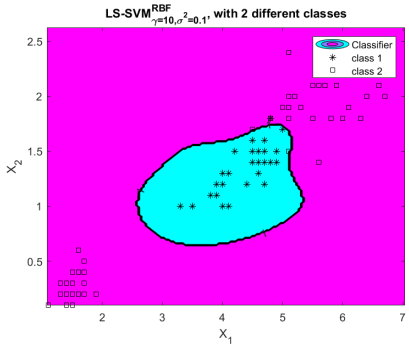

Figure 23: C=10, RBF kernel

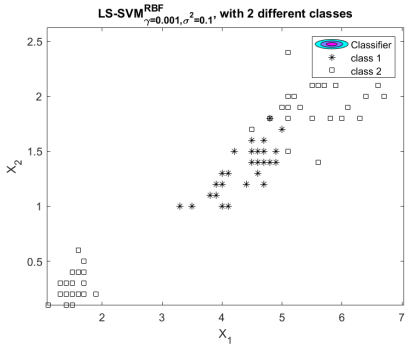

Figure 24: C=0.001, RBF kernel

Q1.3.1.4: Compare your results with SampleScriptiris.m, which is available on Toledo.

The results are surmised in Table 4 below.

SVM Params	Lin C=1	Poly t=1 deg=5	RBF σ^2=0.01	RBF σ^2=0.1	RBF σ^2=1	RBF σ^2=5	RBF σ^2=10	RBF σ^2=25
Test Acc	45	100	90	100	100	100	100	50

Table 4: Table of performance of polynomial kernel for various σ^2 values. C was 1 in all RBF cases

Here out previous conclusions are validated since in the range of 1-10 there are optimal results obtained.

Q1.3.2.1: Compute the performance for a range of gam and sig2 values. Use the random split method, 10-fold crossvalidation and leave-one-out validation. Visualize the results of each method: do you observe differences? Interpret the results: which values of gam and sig2 would you choose?

For each σ^2 and C I tested the values of 1000, 100, 10, 1, 0.1, 0.01, 0.001, once for each validation method, random split, kfold-crossvalidation and leave-one-out. I performed it 3 times in total and the results are displayed below in Figures 25-30. In the cases where one was varied (e.g. C-γ) the other(γ) C was set to 1.

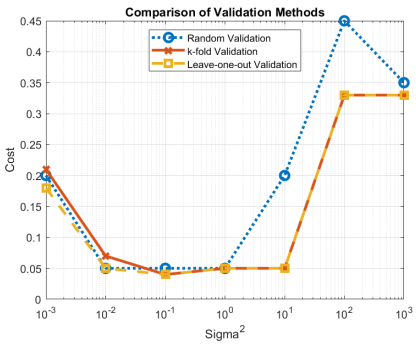

Figure 25: Cost of validation methods Iteration 1, variable σ^2.

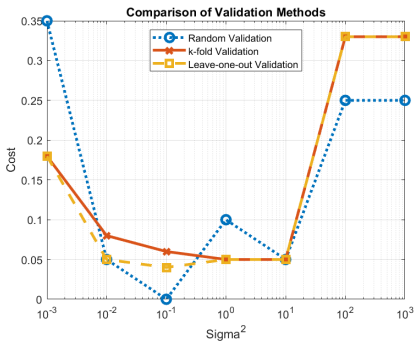

Figure 26: Cost of validation methods Iteration 1, variable σ^2.

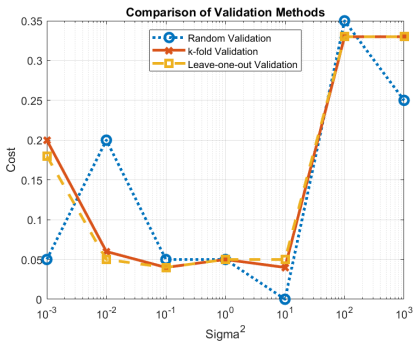

Figure 27: Cost of validation methods Iteration 1, variable σ^2.

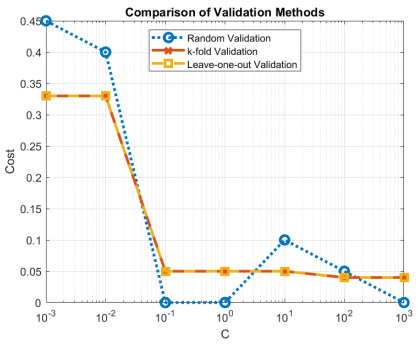

Figure 28: Cost of validation methods Iteration 1, variable C.

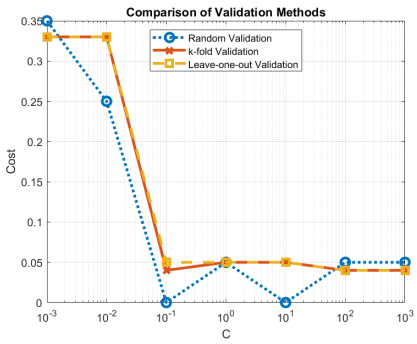

Figure 29: Cost of validation methods Iteration 1, variable C.

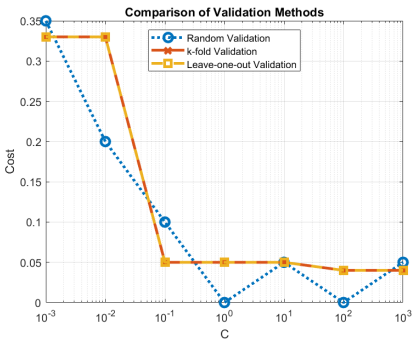

Figure 30: Cost of validation methods Iteration 1, variable C.

Given the above I would pick the C and σ^2 values equal to 1, although the other cases where the leave-one-out and kfold-crossvalidation plateau are worth exploring to broaden that domain. However, due to limited report pages, this will not be done here.

Q1.3.2.2: Why should one prefer crossvalidation over simple validation (random split)? How to choose the value of k in k-fold crossvalidation?

What we can deduce from the above is that:

1. The random split varies wildly between experiments and is thus not dependable as a split.
2. Both the leave-one-out and the kfold-crossvalidation are very robust methods, with the leave-one-out being slight more consistent.

Given the above two, as well as the time complexity of performing the leave-one-out crossvalidation for larger datasets, I would pick the kfold-crossvalidation in every case as the best trade-off between bias-variance and time compexity since in larger datasets, leave-one-out become computationally infeasible. My personal preferences for the value of k vary between 5 and 10 increasing as the dataset decreases in quantity.

Q1.3.3: Try out the different 'algorithm'. What differences do you observe? Why do the obtained hyperparameters differ a lot in different runs? What about the cost? Computational speed? Explain the results.

Search Algorithm	Simplex	GridSearch
C	36.2/71	14200/43800
σ^2	7.70/23.8	43.9/138
Cost	0.0360/0.0052	0.0340/0.0052
Time	0.183/0.045	0.370/0.011

Table 5: Table of performance of the two tuning algorithms for the RBF SVM. 10 iterations were performed. The first number indicates the mean while the second the standard deviation.

I performed 10 iterations for each of the two methods, simplex and GridSearch, and display the results below in Table 4. Although the cost and the time elapsed are very consistent across the 10 experiments we can see that the Simplex and the GridSearch algorithm produce wildly varying results in the values of C and σ^2 as it can be seen by the higher than mean standard deviation in both cases. The difference in cost and speed are due to the inherent differences of the two algorithms. GridSearch is exhaustive, resulting in more search iterations, thus taking it longer to finish, achieving, however, slightly better results. The high difference in parameter tuning could be attributed to the locality of Simplex vs the global nature of the search space of GridSearch, since the former is dependent on initial conditions and can get stuck in local minima, especially in higher dimensional cases.

Q1.3.4.1: In practice, we compute the ROC curve on the test set, rather than on the training set. Why?

We need the model to be able to generalize well on unseen data, in order to make sure it is robust and has not overfitted on the training data. Thus we calculate the ROC curve on the test data.

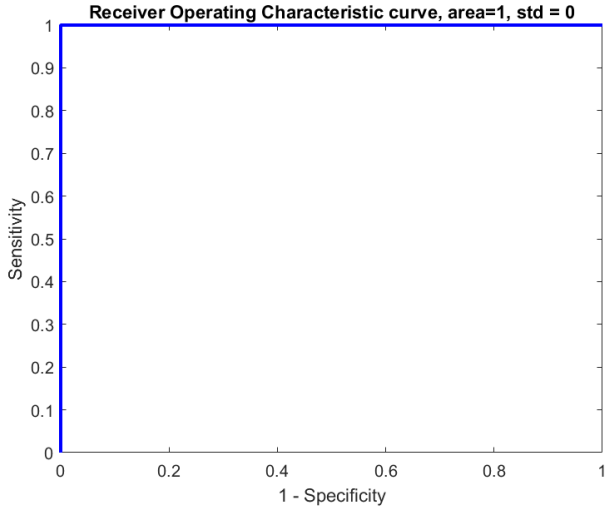

Figure 31: ROC curve of optimised parameters

Q1.3.4.2: Generate the ROC curve for the iris.mat dataset (use tuned gam and sig2 values). Interpret the result.

As we would expect from the previous results the ROC curve (Figure 31) on the test set is perfect using the simplex algorithm, however the dataset available was very limited. (C=1.144, σ^2=0.1285). The ROC curve is exactly the same for the GridSearch optimized parameters (C=103561, σ^2=0.1581).

Q1.3.5.1: How do you interpret the colors of the plot?

The closer to purple, the closer the probability is to 0.5, pink represents areas where the datapoints are highly probable to belong to the one class, while teal represents areas where the datapoints are highly probable to belong to the other. Resulting figures can be seen below in Figures 32-37.

Q1.3.5.2: Change the values of gam and sig2. Visualize and discuss the influence of the parameters on the figure.

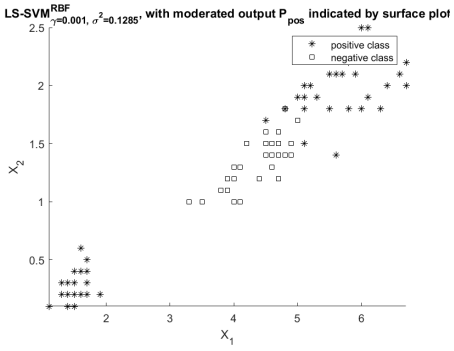

Figure 32: Bayesian Framework. Everything predicted as the negative class.

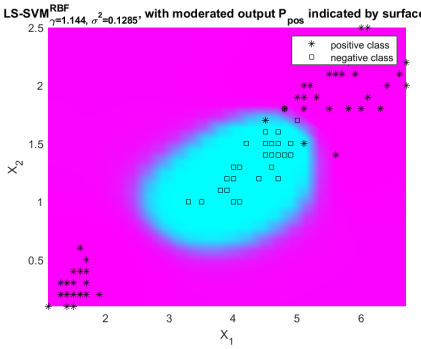

Figure 33: Bayesian Framework. Tuned case.

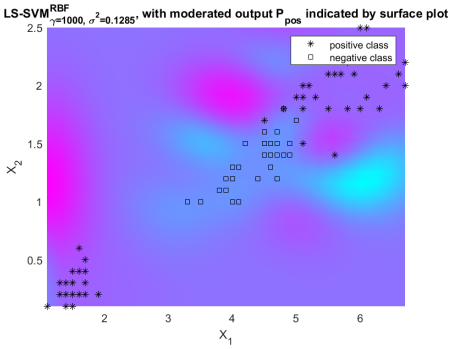

Figure 34: Bayesian Framework. Diffused margins.

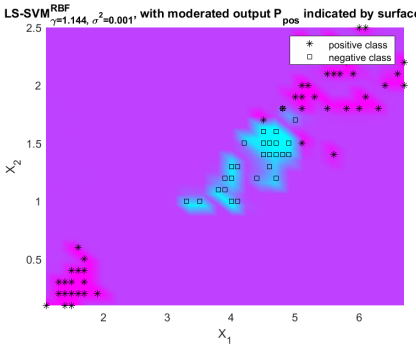

Figure 35: Bayesian Framework. Islets of high certainty.

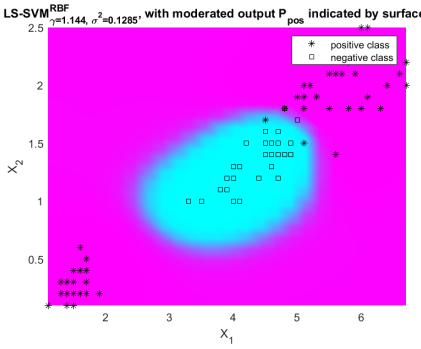

Figure 36: Bayesian Framework. Tuned case.

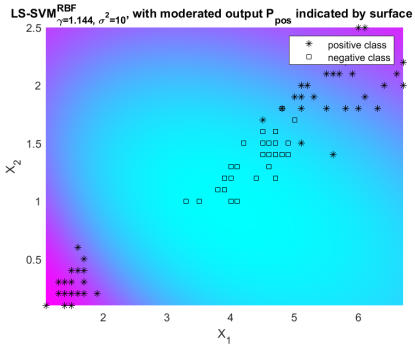

Figure 37: Bayesian Framework. Diffused margins.

In the case of a very low C, the regularization is high and thus the model outputs only the majority class. Increasing C increases the penalization of the classification errors leading first to the tuned version of the model and then to a very diffuse, weird shaped iteration that tries to have no classification error, practically regardless of the margins.

From the above it is easy to recognize the effects of the parameters. As discussed before the parameter σ^2, when too low, gives a small effective radius for the training points, falling off sharply outside the σ^2 area of effect, overfitting. Increasing it produces first the tuned model after which, the increased σ^2, and therefore effective radius, showcases itself as a enlarged teal area to accommodate for the higher datapoint variance in the kernel mapping.

Q2.1: Ripley Dataset

Q2.1.1: Visualize the data. Inspect the data structure: what seems to be important properties of the data? Which classification model do you think you need, based on the complexity of the data?

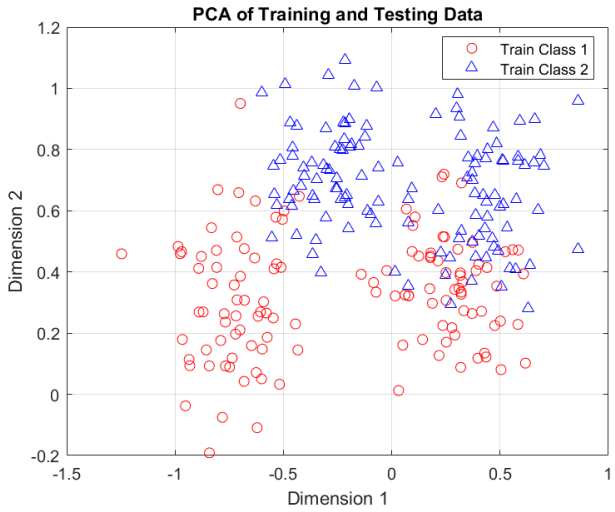

Figure 38: Visualization of training Data. Ripley Dataset.

The Data visualization is displayed below in Figure 38. Based on the data distribution I would think an SVM with a polynomial kernel would be the best to classify the data given the separability of the data of at least degrees 4 to accomodate for the changes in curvature to fit all the data, or a linear one could also fit in a certain angle.

Q2.1.2: Try out different models (linear, polynomial, RBF kernel) with tuned hyperparameter and kernel parameters. Compute the ROC curves. Which model performs best? Which model would you choose?

Experiments were performed using tuning via the gridsearch algorithm for all 3 datasets. ROC curves are displayed below, along with their F1 scores, Accuracies and confusion matrices, since ROC curves are suitable only for balanced data in Figures 39-44.

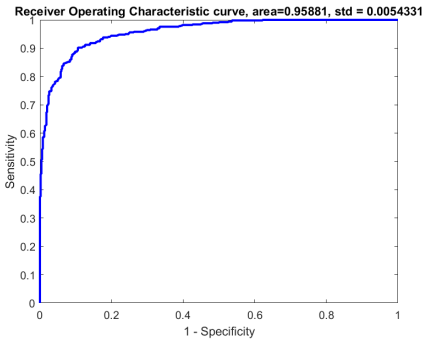

Figure 39: ROC - linear kernel

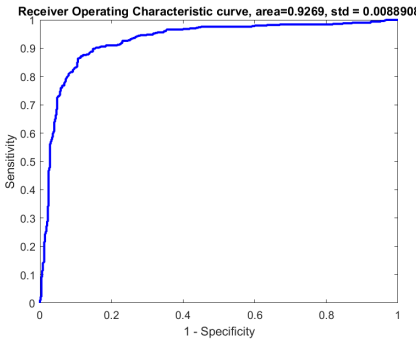

Figure 40: ROC - RBF kernel

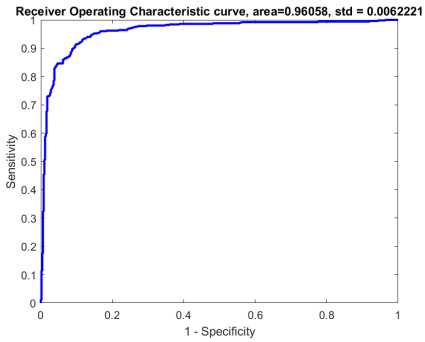

Figure 41: ROC - poly kernel

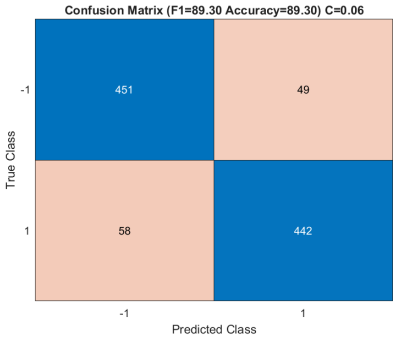

Figure 42: Confusion Matrix linear kernel

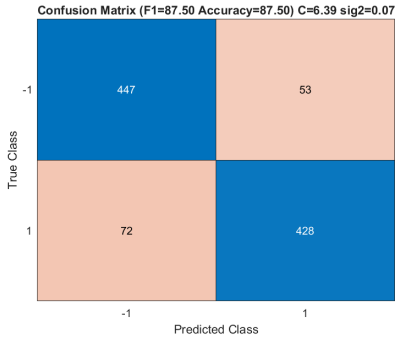

Figure 43: Confusion Matrix RBF kernel

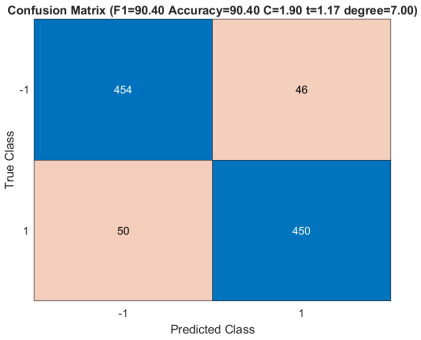

Figure 44: Confusion Matrix poly kernel

Given the dataset is balanced all metrics are suitable for comparison and in all of them the polynomial reigns supreme in ROC AUC, in accuracy and f1

Q2.1.3: Are you satisfied with the performance of your model? Would you advise another methodology?

I am overall satisfied with the performance of the model, especially given that the train set was bigger than the test set. With more data I think the model could become more robust.

Q2.2: Breast Dataset

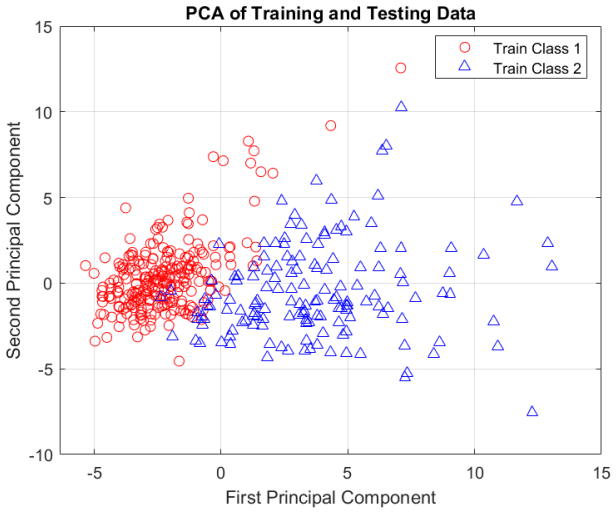

Figure 45: Visualization of training Data. Ripley Dataset.

Q2.2.1

The Data visualization is displayed below in Figure 45. The plot was produced by utilizing PCA keeping the components that explained 95% of the variance. Those components were 10 out of 30. Based on the data distribution I would think an SVM with a polynomial kernel would be the best to classify the data given the separability of the data of maximum 2 degrees to accommodate for the slight changes in curvature to fit all the data but I think the linear SVM would perform relatively well also. Given the limited visualization of the 2 out of the 10 principal components the RBF kernel could be also the solution.

Q2.2.2

ROC curves are displayed below, along with their F1 scores, Accuracies and confusion matrices, since ROC curves are suitable only for balanced data in Figures 46-51.

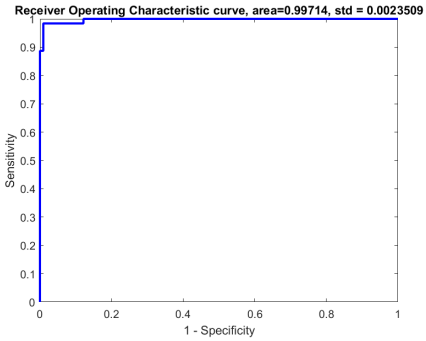

Figure 46: ROC - linear kernel

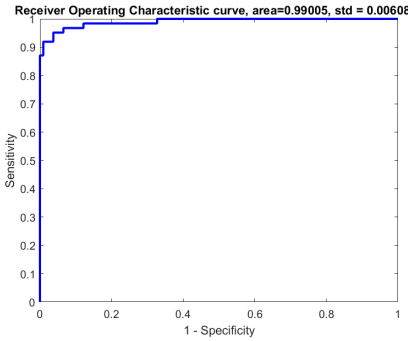

Figure 47: ROC - RBF kernel

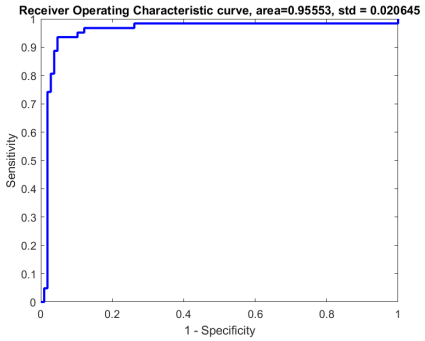

Figure 48: ROC - poly kernel

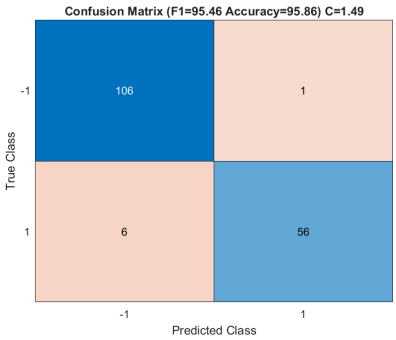

Figure 49: Confusion Matrix linear kernel

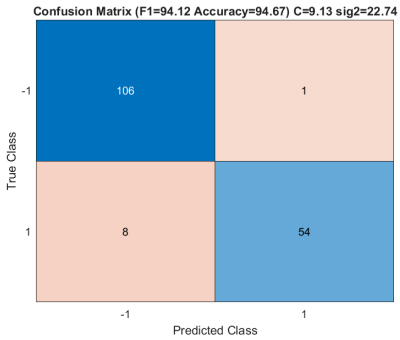

Figure 50: Confusion Matrix RBF kernel

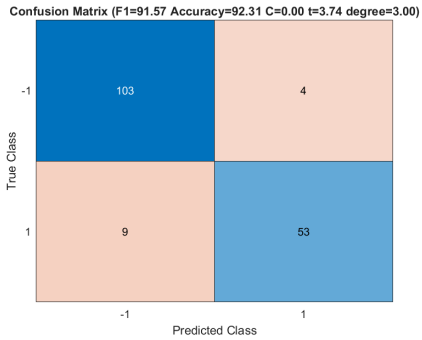

Figure 51: Confusion Matrix poly kernel

Given the dataset is unbalanced the main metric utilized to compare the models is the F1 score although all of them concur that the linear model is the best in this case.

Q2.2.3

I am overall satisfied with the performance of the model. Given more time, and mainly space, I would also compute the precision recall curves and use them as an even more robust metric.

Q2.3: Diabetes Dataset

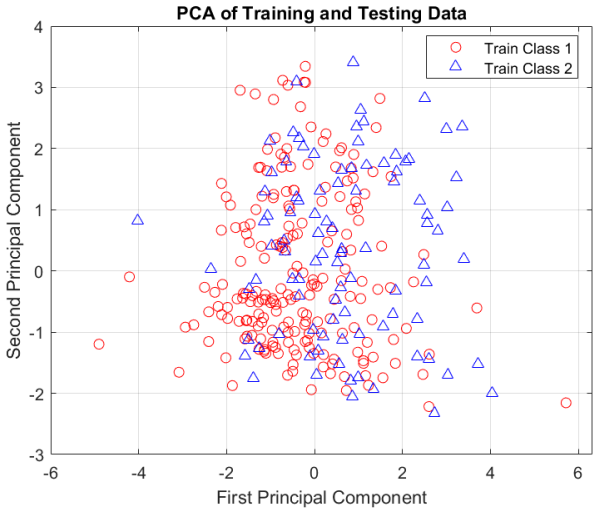

Figure 52: Visualization of training Data. Ripley Dataset.

Q2.1.1

The Data visualization is displayed below in Figure 52. The plot was produced by utilizing PCA keeping the components that explained 95% of the variance. Those components were 7 out of 8. Based on the data distribution I would think an SVM with a RBF kernel would be complex enough to accommodate for this case of "messy" clustering of the data.

Q2.1.2

ROC curves are displayed below, along with their F1 scores, Accuracies and confusion matrices, since ROC curves are suitable only for balanced data in Figures 53-58.

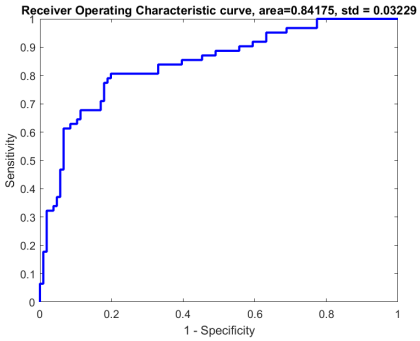

Figure 53: ROC - linear kernel

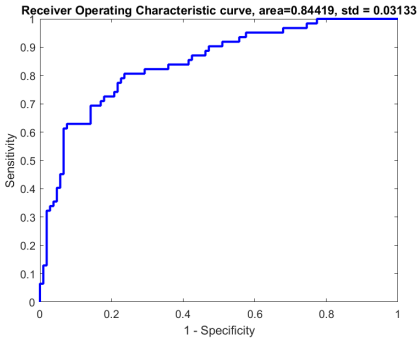

Figure 54: ROC - RBF kernel

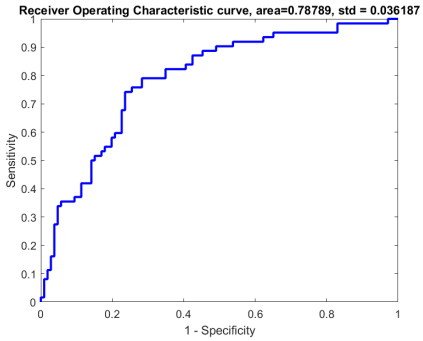

Figure 55: ROC - poly kernel

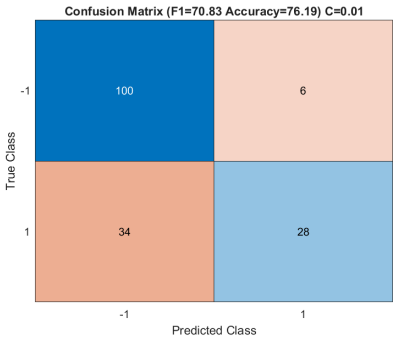

Figure 56: Confusion Matrix linear kernel

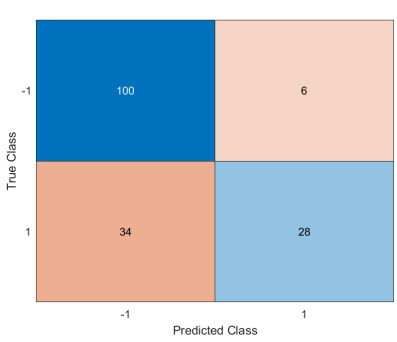

Figure 57: Confusion Matrix RBF kernel

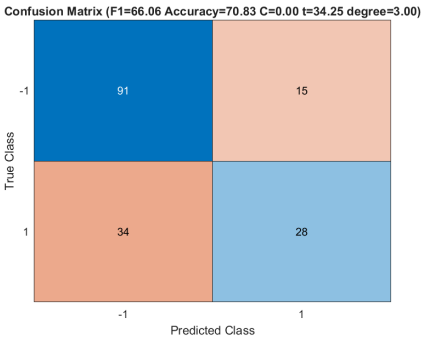

Figure 58: Confusion Matrix poly kernel

Given the dataset is unbalanced the main metric utilized to compare the models is the F1 score although all of the metrics prove that the linear and the RBF kernel produce similar results in this case.

Q2.1.3

I am overall satisfied with the performance of the model. Given more time, and mainly space, I would do as mentioned above and also compute the precision recall curves and use them as an even more robust metric.

Second Exercise Session

Q1.1.1: Construct a dataset where a linear kernel is better than any other kernel (around 20 data points). What is the influence of e (try small values such as 0.10,0.25,0.50,...) and of Bound (try larger increments such as 0.01,0.10,1,10,100). Where does the sparsity property come in?

The equation for the SVR is:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*)$$

subject to

$$\begin{aligned} & y_i - \langle w, x_i\rangle - b \leq \epsilon + \xi_i, \\ & \langle w, x_i\rangle + b - y_i \leq \epsilon + \xi_i^*, \\ & \xi_i, \xi_i^* \geq 0 \end{aligned} \tag{8}$$

What the parameter e (ϵ above) does, is basically determine the width of the regressor line inside of which the datapoints are not accounted for errors (Technically since we are in higher dimensions e represents the radius of a hypercylinder). If the slack variables for above and below the regressor line are inside the width e, then their errors are not taken into account. The bound is none other than the regularization term C. Increasing it increases the cost of miss-classification at the expense of generalization (bias-variance trade-off).

The term e is related to the sparseness property, producing it since all points inside this margin are not taken into account in the calculation of the cost function above. C is inversely correlated with sparseness, on the other hand. Increasing C makes the model fit the trading data more closely. This induces more support vectors. The reverse holds for lower values of C. Below in Figures 59-61 I showcase the differences when changing the bound C and in Figures 62-65 when changing the term e. Note that for $C > 1$ and e=0.15 the plots remained the same.

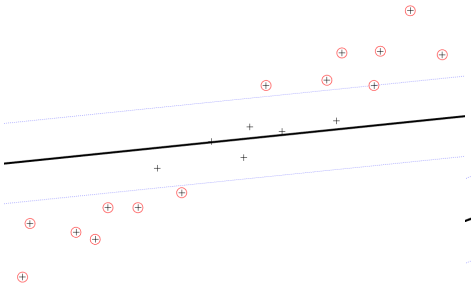

Figure 59: C=0.01 e=0.15.

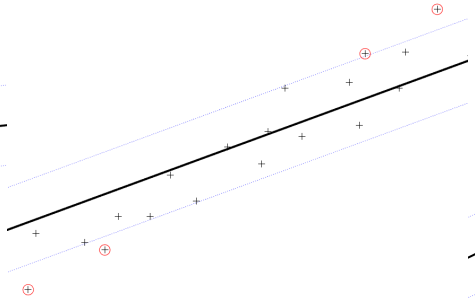

Figure 60: C=0.1 e=0.15.

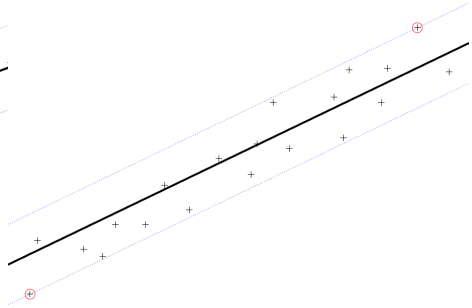

Figure 61: C=1 e=0.15.

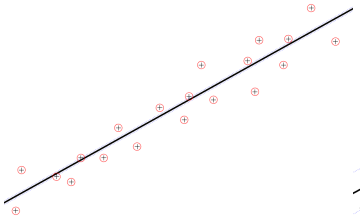

Figure 62: C=1 e=0.01.

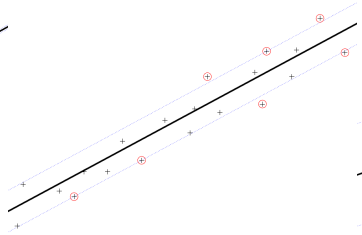

Figure 63: C=1 e=0.1.

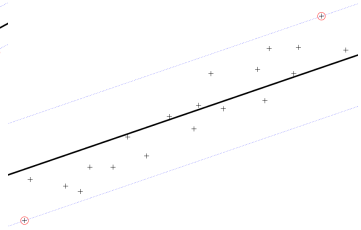

Figure 64: C=1 e=0.25.

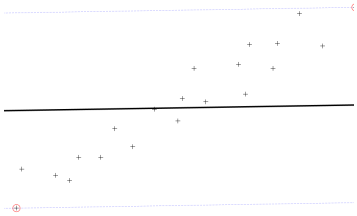

Figure 65: C=1 e=0.5.

Q1.1.2: Construct a more challenging dataset (around 20 data points). Which kernel is best suited for your dataset? Motivate why.

Given a more spread-out dataset, trying the same linear kernel with certain specific parameters such as C=1, e=0.15, as well as another as the RBF with the same parameters, yields better results for the nonlinear kernel since it can adjust its curvature to better handle outlying datapoints and minimize the cost function (Figures 66-67 below).

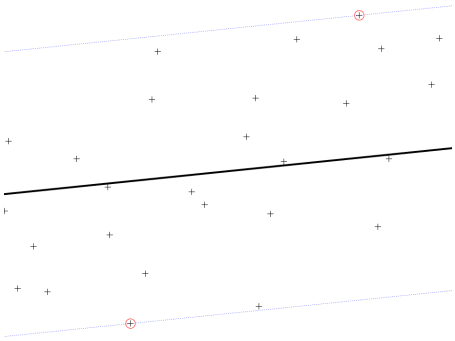

Figure 66: C=1 e=1.

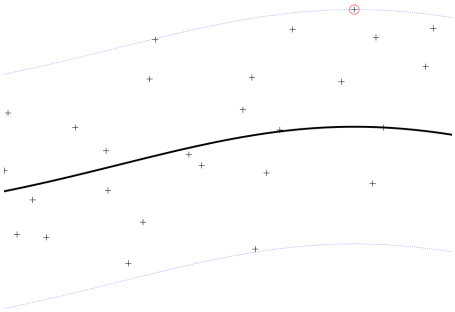

Figure 67: C=1 e=1.

Q1.1.3: In what respect is SVM regression different from a classical least squares fit?

The main differences I would say are three.

1. **Kernel trick**: allows the SVR to robustly handle non-linear datasets.
2. **Hyperparameters**: tuning them allows to reach the desired qualities when training a model.
3. **e**: allows the model to not penalize certain datapoints whereas the least squares penalizes all points.

Q.1.2.1-2: Try out a range of different gam and sig2 parameter values (e.g., gam = $10, 10^3, 10^6$ and sig2 = 0.01,1,100) and visualize the resulting function estimation on the test set data points. Discuss the resulting function estimation. Report the mean squared error for every combination (gam, sig2). Do you think there is one optimal pair of hyperparameters? Argument why (not).

Seeing the Figures 70-78 it can be seen that the combination of a high $C = 10^6$ along with a medium $\sigma^2 = 1$ is the best combination (Figure 77). It fits both the training and testing data very well performing with the best MSE. Increasing sigma results in underfitting while decreasing results in overfitting. Apart from the metric measured, which gives very good results even in the case of overfitting, a qualitative look at the estimated function is necessary to ensure that it properly resembles the true one. In case the true function is not known I would consider a good rule that smoother functions are generally better performing given more data, since they account better for the presence of noise. Even if the resulting metric was worse in Figure 77 than in Figures 70, 73 or 76, I would still pick that one as the best.

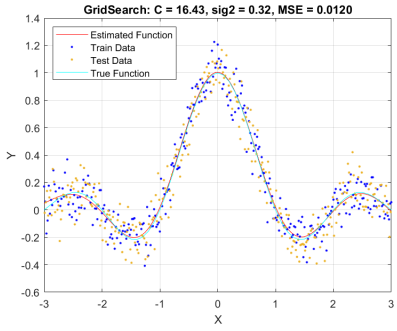

Figure 68: GridSearch Tuning.

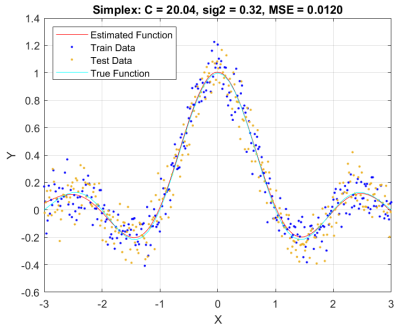

Figure 69: Simplex Tuning.

Q1.2.1.3: Tune the gam and sig2 parameters using the tunelssvm procedure. Use multiple runs: what can you say about the hyperparameters and the results? Use both the simplex and GridSearch algorithms and report differences.

We can see that both algorithms generally produce similar results in both cases. Given that the underlying function is very smooth this is expected. It would be highly interesting in seeing how these algorithms perform in multidimensional functions where I think the exhaustive nature of GridSearch would shine.

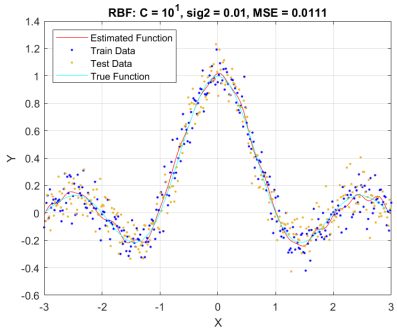

Figure 70: C=10, σ^2=0.01.

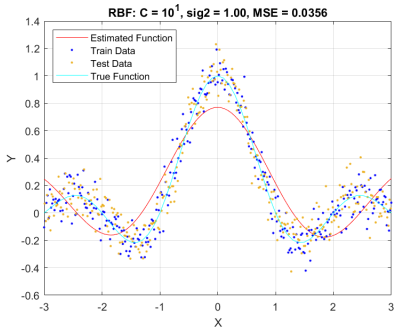

Figure 71: C=10, σ^2=1.

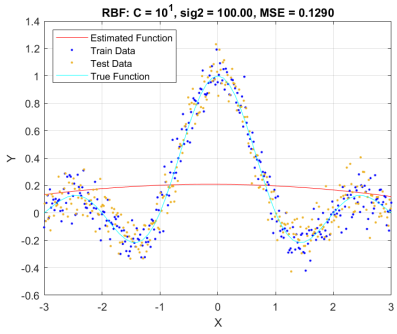

Figure 72: C=10, σ^2=100.

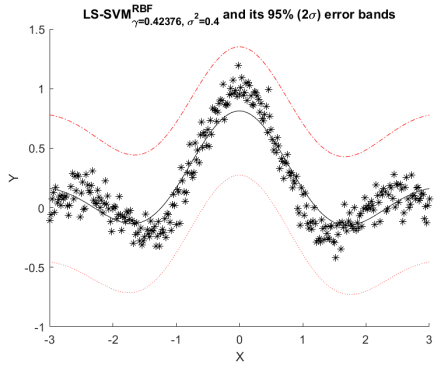

Figure 79: GridSearch Tuning.

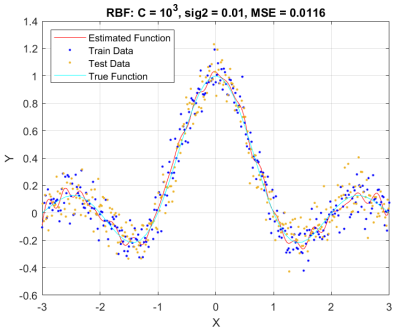

Figure 73: $C = 10^3, \sigma^2$=0.01.

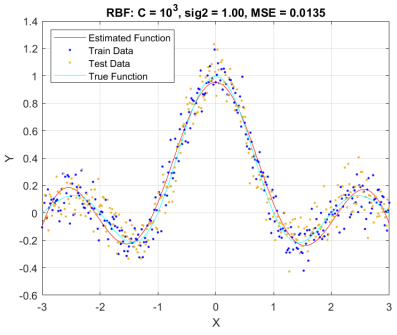

Figure 74: $C = 10^3, \sigma^2$=1.

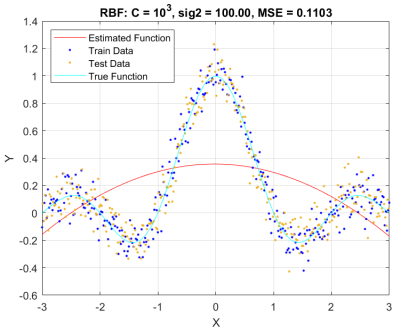

Figure 75: $C = 10^3, \sigma^2$=100.

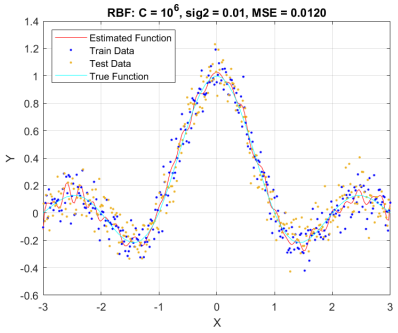

Figure 76: $C = 10^6, \sigma^2$=0.01.

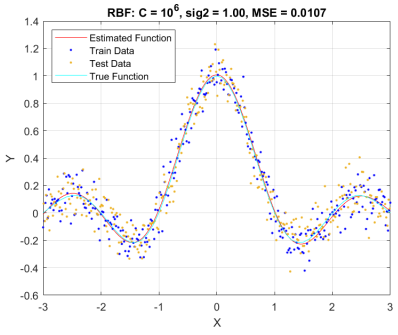

Figure 77: $C = 10^6, \sigma^2$=1.

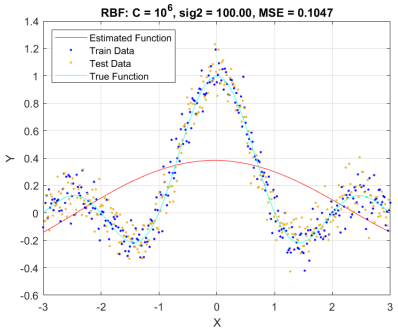

Figure 78: $C = 10^6, \sigma^2$=100.

Q1.2.2: Discuss in a schematic way how parameter tuning works using the Bayesian framework. Illustrate this scheme by interpreting the function calls denoted above.

In the Bayesian Framework of Gaussian Processes, in the context of LS-SVM, in order to minimize the cost function of the LS-SVM, the assumption is made is that all data points will be sampled from a Gaussian Distribution of (N+M) where N are the train and M the test points with a prior of zero means and variances that are approximated by a Gaussian Kernel which is used to approximate the true covariance function. The process goes as follows for the parameters of the models first, then the hyperparameter C and then the hyperparameter σ^2 of the covariance matrix (RBF kernel).

1. An initial estimate of C (=10 in our case) and σ^2 (=0.4) is made.
2. A new data point is measured and the posterior is created as the probability of getting this data point conditioned on prior observations, based on Bayes' theorem.
3. The cost function is calculated and a new parameter value is picked for the parameter we are trying to optimize. This is done by optimizing other functions with gradient-based methods.
4. Steps 2,3 are iterated until all datapoints are exhausted.
5. Steps 1-4 are repeated with the prior of the next Step 1 being the posterior at Step 4 of the previous iteration if/until convergence is reached.

First this is done for the model parameters, then it is done for C to get the correct trade-off between bias and variance as discussed earlier, and lastly, it is done on the basis of the kernel parameter to find an optimal value.

The commands bay_lssvm are used to calculate the cost functions when optimizing for the model (used with the number 1), C (used with 2) or σ^2 (used with 3), while the commands bay_optimize are utilized to find the same optimal parameters (spoiler alert: the sig2 is found to be the same with a C=0.4625). Lastly, the bay_errorbar command creates a plot of the function along with a 2σ interval in either side.

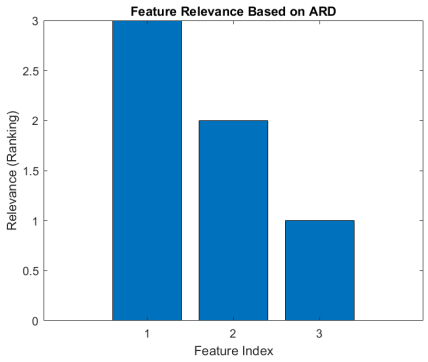

Figure 80: Ranking on the 3 dimensional data.

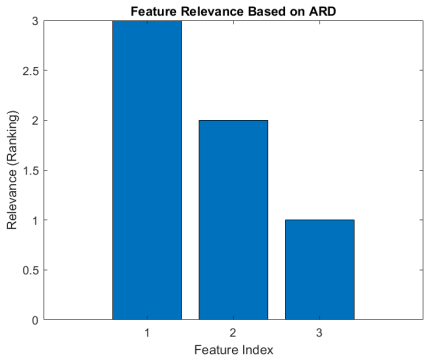

Figure 81: Ranking on the 3 dimensional data.

Q1.3.1: Automatic Relevance Determination. Visualize the results in a simple figure.

In Figure 80 the results are displayed. Since the label values are directly a function from the first dimension of the train data the only logical conclusion would be that the first dimension is the most relevant, while for the other two since the training data is random, any one could randomly be selected as the second or third most relevant.

Q1.3.2: How can you do input selection in a similar way using the crossvalidate function instead of the Bayesian framework?

You can perform crossvalidation, separately for each input dimension by tuning the LSSVM each time and assessing the results using MSE, the value of the cost function and qualitative aspects such as the figures. In our examples below the associated costs were 0.012, 0.15 and 0.15 (Figures 82-84).

Q1.4.1: Robust Crossvalidate: Visualize and discuss the results. Compare the non-robust version with the robust version. Do you spot any differences?

Yes, there are stark differences, in Figure 85 where we have used the typical crossvalidation, the outliers have a significant impact, especially in the first and third peaks. However, in the case of the robust (weighted) LS-SVM this is much less pronounced, especially in the third peak since the robust estimates of the outliers fall with the inverse of the distance whereas for the outliers in the first peak the outliers have a noticeable but less pronounced result.

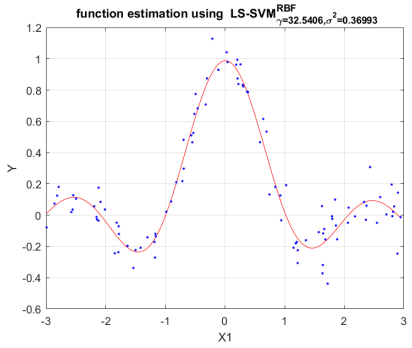

Figure 82: Results for 1^{st} dimension.

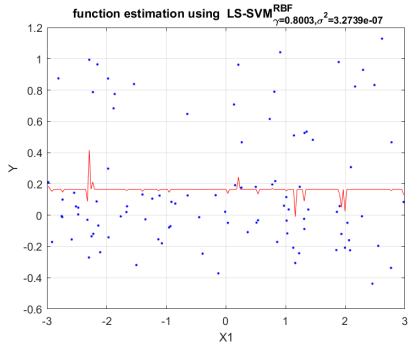

Figure 83: Results for 2^{nd} dimension.

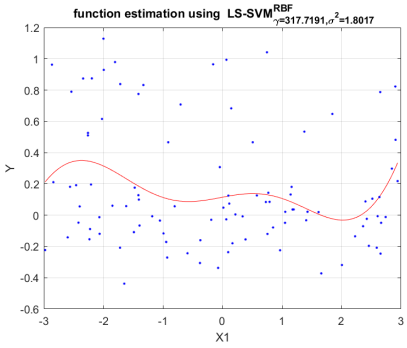

Figure 84: Results for 3^{rd} dimension.

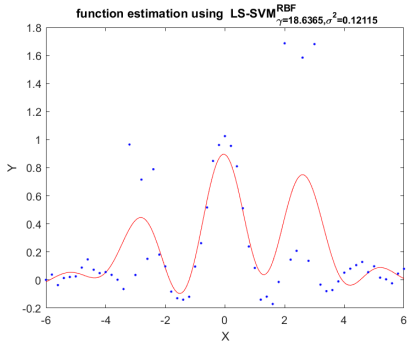

Figure 85: Typical crossvalidation.

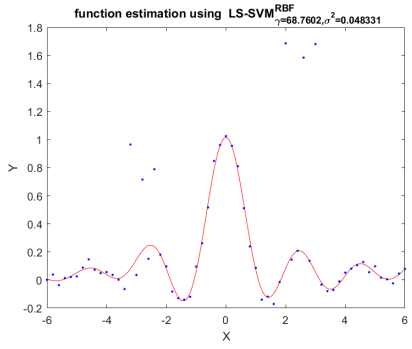

Figure 86: Robust crossvalidation.

Huber weighting

$$w(r) = \begin{cases} 1, & \text{if } |r| \leq b, \\ \frac{b}{|r|}, & \text{if } |r| > b. \end{cases}$$

β being a constant

Q1.4.2: Why in this case is the mean absolute error ('mae') preferred over the classical mean squared error ('mse')?

Comparing the two equations:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|, \qquad \text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

It can be easily seen that in the case of outliers from the model, their respective term in the case of MSE is higher than in MAE, giving us a more false estimation of the performance of our model by taking these outliers more into account. Knowing that the data is corrupted by outliers allows us to accommodate for it with MAE.

Q1.4.3: Try alternatives to the weighting function wFun (e.g., 'whampel', 'wlogistic' and 'wmyriad'. Report on differences.

The results for all 4 techniques are displayed below in Figures 87-90 where we can see that hampel is the best fitting one followed by the myriad, then the huber and last comes the logistic weighting. Looking at the weighting equations of each case

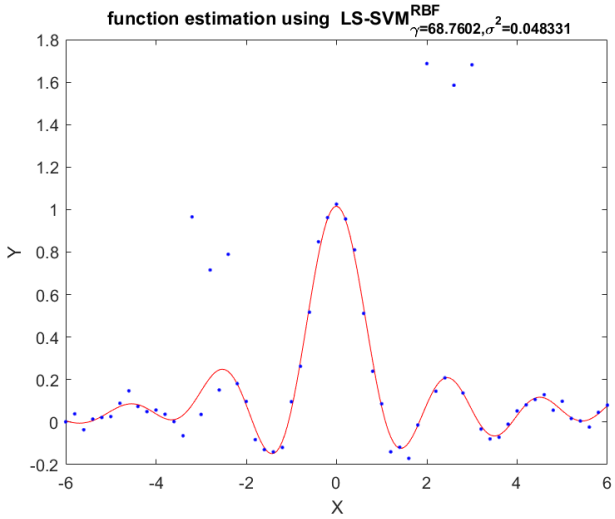

Figure 87: huber weighting

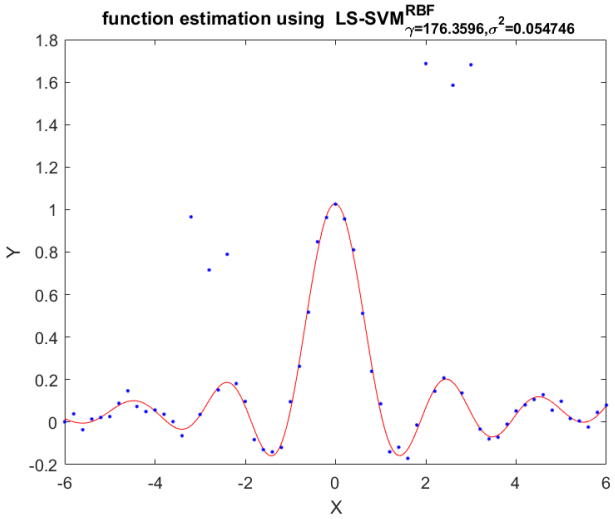

Figure 89: hampel weighting

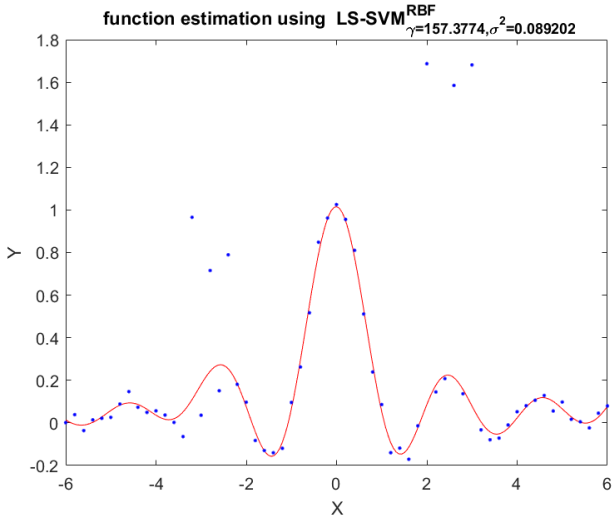

Figure 88: logistic weighting

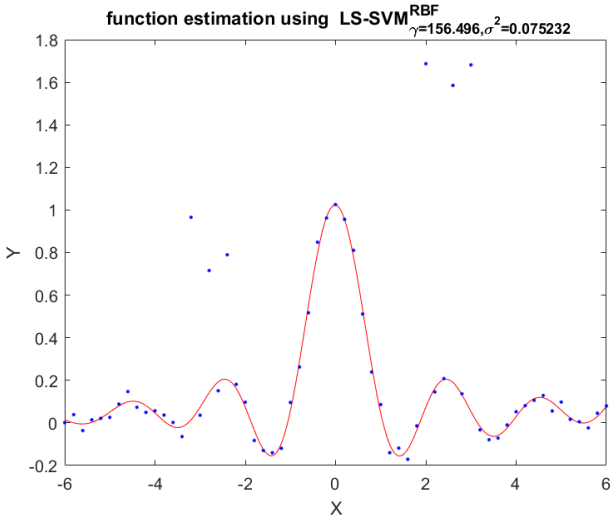

Figure 90: myriad weighting

Logistic weighting

$$W_z = \frac{tanh(r)}{r}$$

Hampel Weighting

$$W(r) = \begin{cases} 1 & \text{if } |r| \le b_1, \\ \frac{(b_2 - |r|)}{(b_2 - b_1)} & \text{if } b_1 < |r| \le b_2, \\ 0 & \text{if } |r| > b_2 \end{cases}$$

b_1, b_2 being constants.

Myriad weighting

$$W(r) = \frac{\delta^2}{\delta^2 + r^2}$$

δ being a constant

The above results are to be expected since Hampel weighting sets to 0 the influence of outliers, given that you know which are so, else you run the risk of removing noisy data. The Myriad in inversely proportional to the square of the distance so comes in second, while the Huber inversely proportional to the distance, making its use proper when there is not sure knowledge that a data is an outlier or just part of the function, while logistic sacrifices the robustness for differentiability and more smooth weight decay.

Logmap Dataset

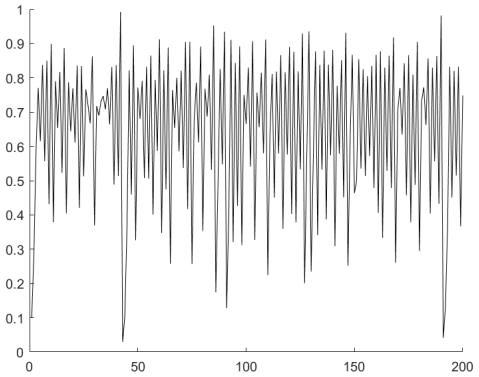

Figure 91: All logmap data.

Q.2.2: As indicated numerous times before, the parameters gam and sig2 can be optimized using crossvalidation. In the same way, one can optimize order as a parameter. Define a strategy to tune these 3 parameters. Do time series prediction using the optimized parameter settings. Visualize your results. Discuss.

I performed a gridsearch for orders of value, 3, 5, 10, 15, 20, 25, 30, 50, 100. I used the Hampel weighting scheme for crossvalidation using gridsearch and got the following results displayed below in Figures 90-98. Figure 99 contains the best results plotted also against the Noiseless Data with its MAE computed for the Noiseless Data.
The reasons behind picking this specific value of parameters is the same as with the sinusoid function. The distribution of the data was obvious that they had a period of 2 time points and it was apparent that there were three at least outliers. Therefore, I chose these values since they showed a good MAE in addition to ignoring these outliers and robustly maintaining the same frequency.

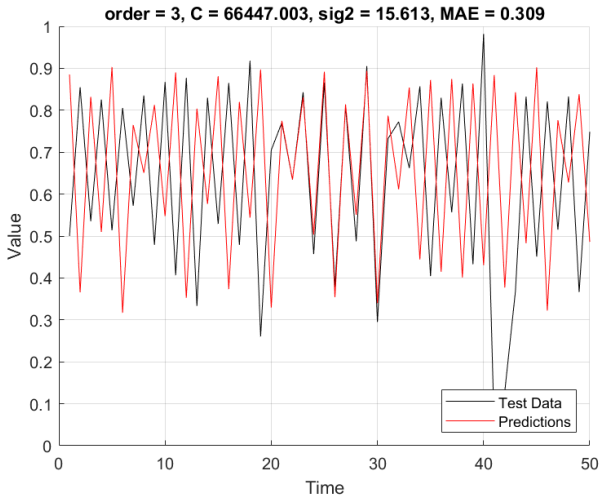

Figure 92

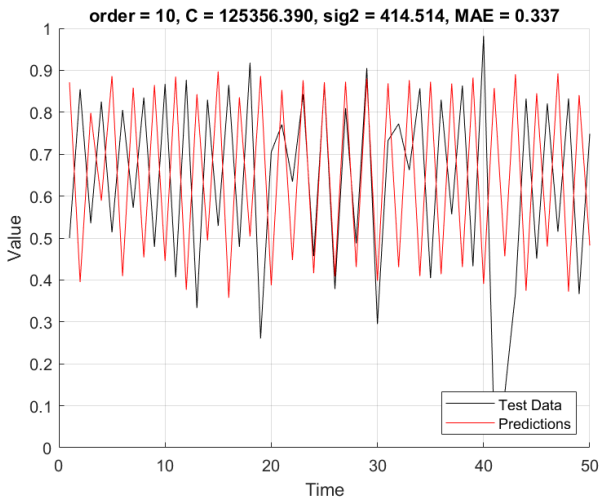

Figure 93

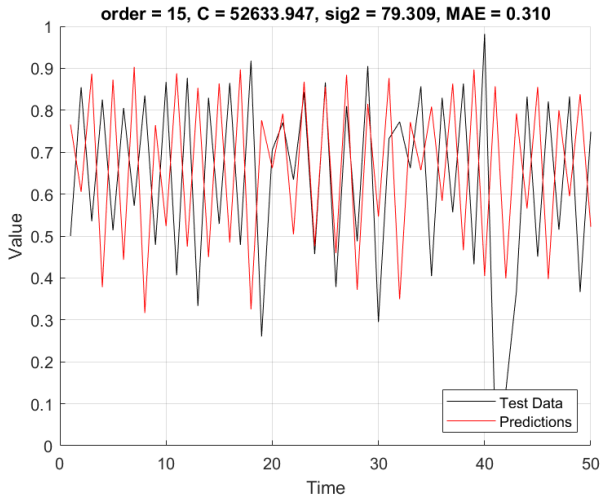

Figure 94

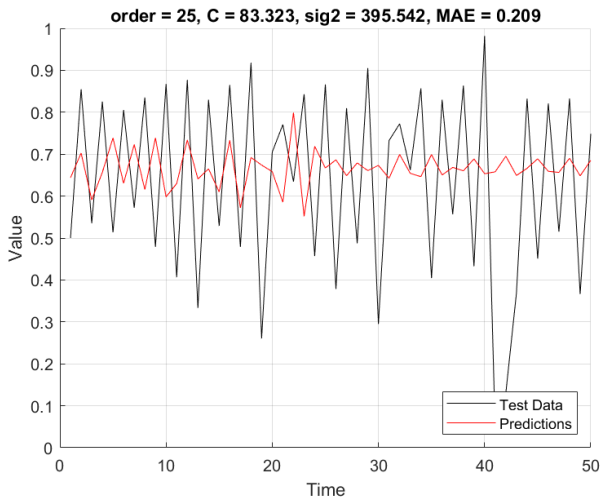

Figure 95

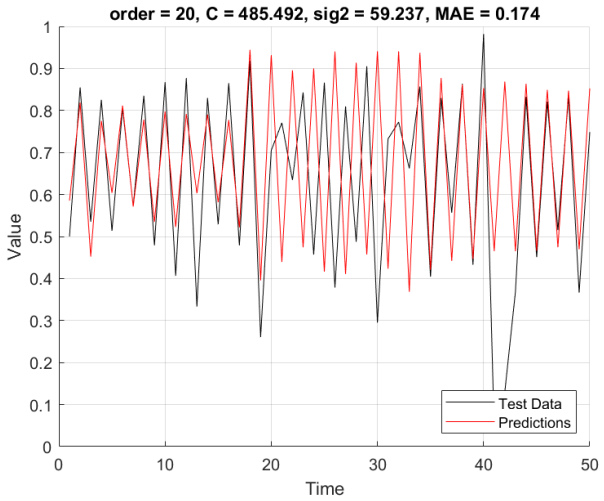

Figure 96

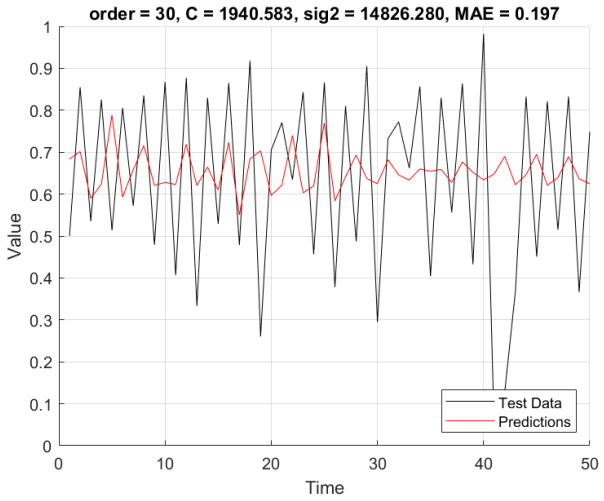

Figure 97

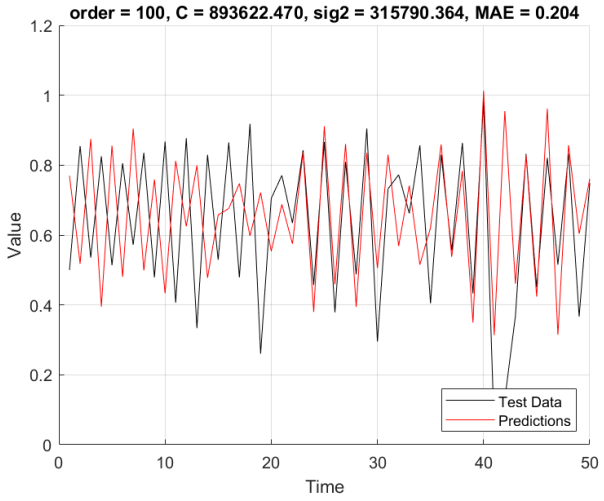

Figure 98

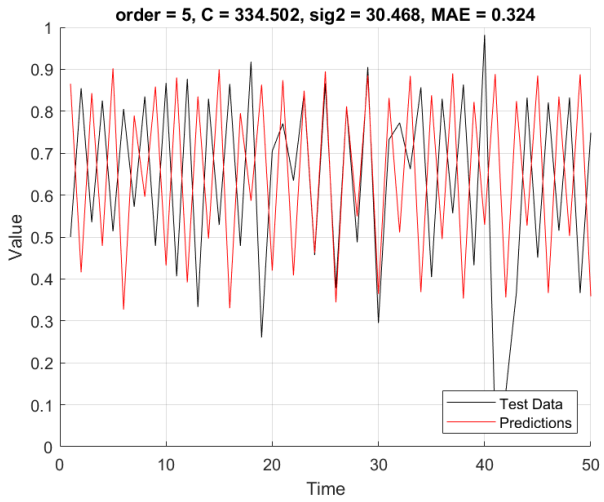

Figure 99

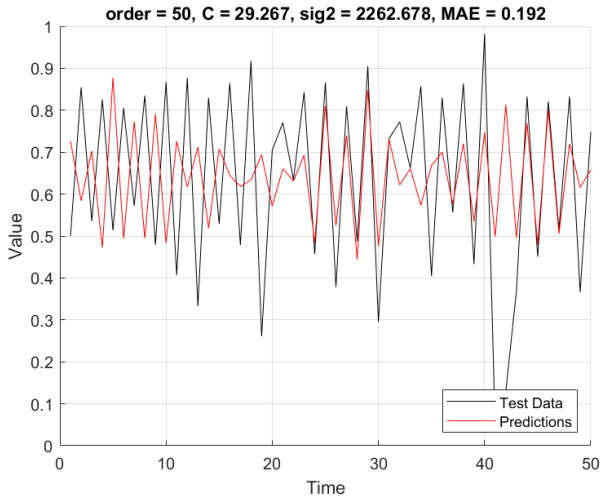

Figure 100

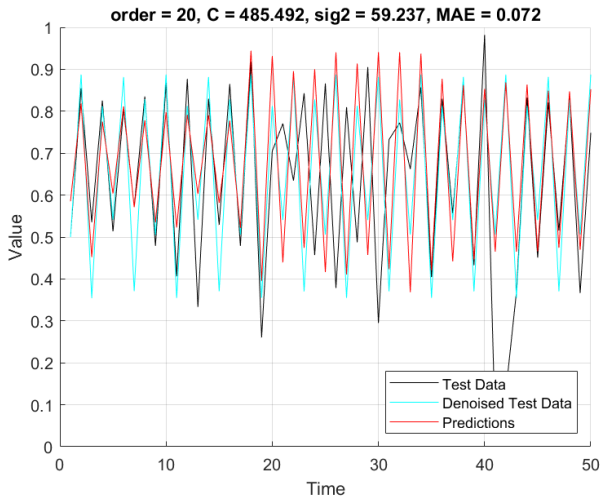

Figure 101: Best Results with Noiseless Data. Order=20, MAE=0.006.

Santa Fe dataset

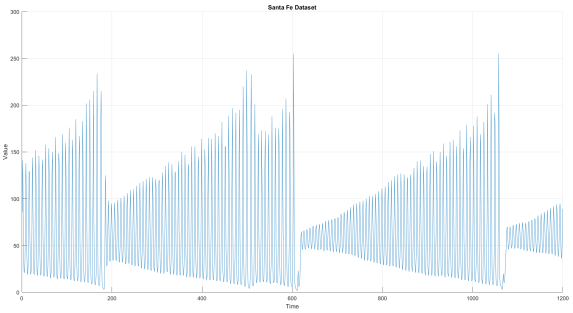

Figure 102: All Santa Fe data.

Q.2.3.1-2: Does order = 50 for the utilized autoreressive model sounds like a good choice? Would it be sensible to use the performance of this recurrent prediction on the validation set to optimize hyperparameters and the model order?

This time the data is very different, the system has a much higher period (Figure 102) and is much more time-dependent. Due to this, I think the model will need a higher order of magnitude than before, in order to contain enough data to grasp well enough the intricacies of the time series, so I think a value of order 50 will not be enough and we would need a different cross-validation scheme to account for such a behavior, since splitting the time series into n parts and concatenating the n-1 will mess with the time dependency. However, visualizing the results in Figure 103-111 using the previous scheme, it can be seen they are not bad in certain cases such as orders 50, 75 and 100. The orders tested were 5, 10, 25, 50, 75, 100, 150, 200, 250.

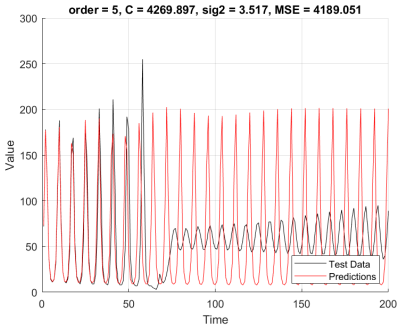

Figure 103

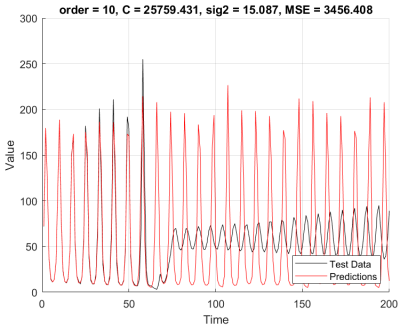

Figure 106

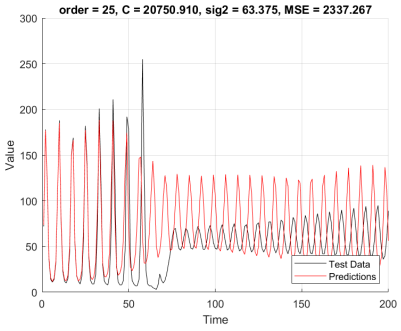

Figure 109

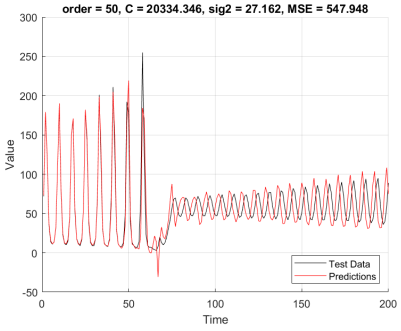

Figure 104

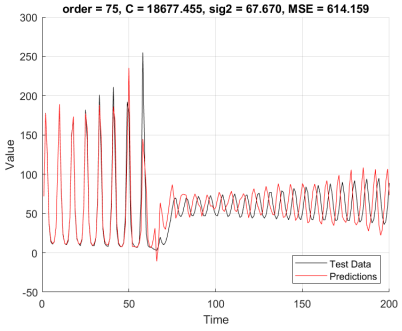

Figure 107

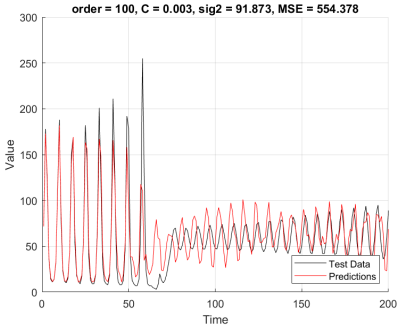

Figure 110

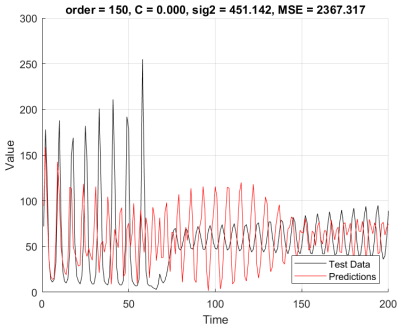

Figure 105

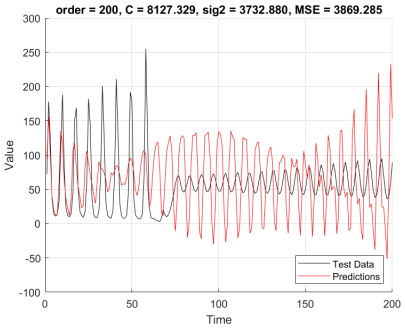

Figure 108

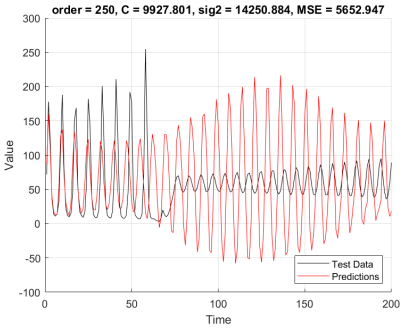

Figure 111

Tune the parameters (order, gam and sig2) and do time series prediction. Visualize your results. Discuss.

For the reasons above I split the dataset using the tpartition function into 5 sets and used the last one with the most data points for validation with the same order list as above. The results are displayed below in Figures 112-120. We can see the best results are for order 5 and 10 and when comparing with the method performed above we get a much better performance with an improvement of 5 times.

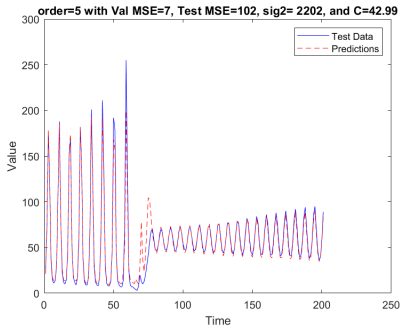

Figure 112

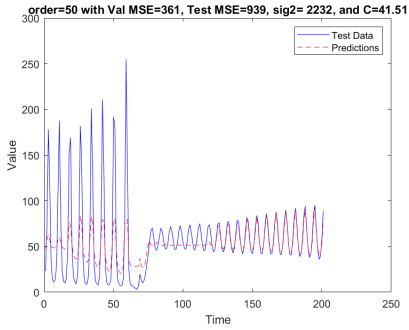

Figure 113

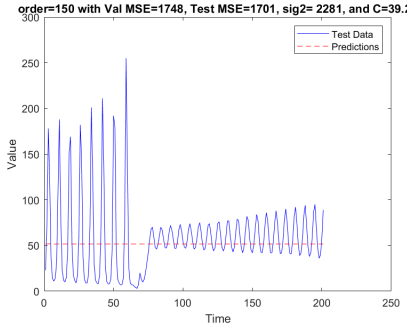

Figure 114

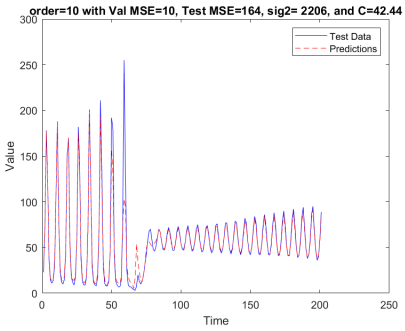

Figure 115

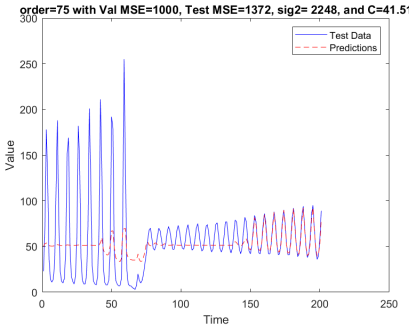

Figure 116

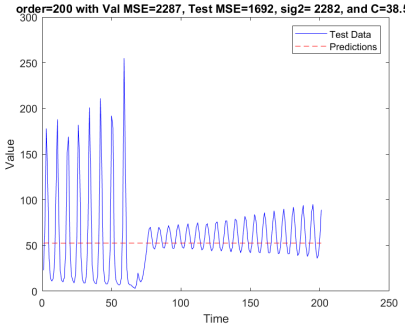

Figure 117

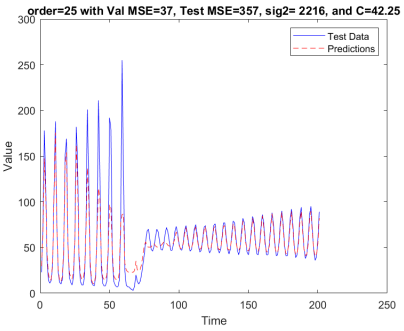

Figure 118

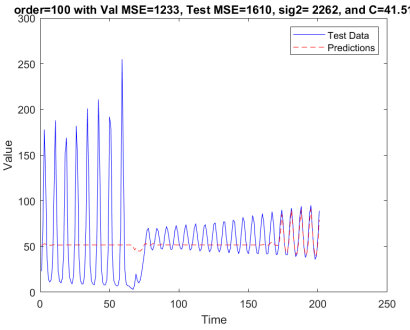

Figure 119

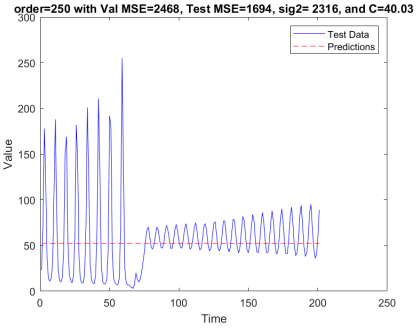

Figure 120

Third Exercise Session

Q1.1.1: Describe how you can do denoising using PCA. Describe what happens with the denoising if you increase the number of principal components.

In PCA, you calculate the covariance matrix of the centered-in-their-means data, computing the eigenvalues and eigenvectors of that matrix, effectively creating an orthonormal vector space, dimensionally equal to the original where we can project our data. The eigenvalues' magnitude represents the percentage of the variance that is explained along each orthonormal eigenvector.

Denoising using the above is based on the principle that random noise will be spread relatively evenly across all principal components and thus will impact less the high variance/meaningful components. Thus, by finding the elbow point in the plot of variance to eigenvalue index where variance starts diminishing rapidly, we can investigate around that eigenvalue index and pick the one that reduces most the noise. Continuing to increase the principal components we will project our data on will introduce more noise than the optimal point.

Q1.1.2: Compare linear PCA with kernel PCA. What are the main differences? How many principal components can you obtain?

The differences are summarized for simplicity, clarity and user-readability in the table 6 below assuming n data-points with p input features with PC notating principal components:

Table 6: PCA vs Kernel PCA

	PCA	**Kernel PCA**
Type of Data	Linear	Non-Linear
Num. of PCs	min(n,p)	$\leq n$
Computational Cost	Low	High
Memory Cost	Low	Very High
PC derivation via	Covariance Matrix	Kernel Matrix

The typical PCA performs only linear operations on the data, calculating their pxp covariance matrix and extracting the eigenvalues and eigenvectors from it. In contrast, kernel PCA uses the kernel trick, utilizes kernel functions, to map the data from $\mathbb{R}^p \to \mathbb{R}^M$ where M > p. Kernel PCA utilizes the kernel Matrix in order to calculate the principal components since the calculation of the MxM is computationally infeasible. That is why while PCA can achieve a number of principal of min(n,p), kernel PCA achieves components up to n. So kenrel PCA can handle non-linear data much better. Also, by having a higher number potentially of components that are dependent on the number of samples, it is not constrained when the data is composed of a small number of dimensions. For example, having a dataset comprised of 100 samples with two dimenions, makes the PCA able to give maximally 2 principal components while kernel PCA could gibe maximally 100. It comes all with the cost of higher memory and computational cost

The reason that the approximation of the covariance matrix by the kernel matrix is feasible is because the eigenvectors of the covariance matrix can be approximated by weighted a sum of the kernel function. The weights of the kernel functions are also the eigenvectors of the kernel matrix. The mathematical derivation given covariance matrix C, its eigenvalues λ, eigenvectors v and kernel function $\phi(x)$ is:

$$\mathbf{C}\mathbf{v} = \frac{1}{N}\sum_{i=1}^{N} \phi\left(\mathbf{x}^{(i)}\right)\phi\left(\mathbf{x}^{(i)}\right)^{\top}\mathbf{v} = \lambda\mathbf{v} \tag{9}$$

$$\implies \mathbf{v} = \sum_{i=1}^{N} \underbrace{\frac{\phi\left(\mathbf{x}^{(i)}\right)^{\top}\mathbf{v}}{N\lambda}}_{\alpha_i}\phi\left(\mathbf{x}^{(i)}\right) \tag{10}$$

substituting the above into the definition of the covariance matrix we get:

$$\lambda\sum_{j=1}^{N}\alpha_j\phi\left(\mathbf{x}^{(j)}\right) = \lambda\mathbf{v} = \mathbf{C}\mathbf{v} = \frac{1}{N}\sum_{i=1}^{N}\phi\left(\mathbf{x}^{(i)}\right)\phi\left(\mathbf{x}^{(i)}\right)^{\top}\left(\sum_{j=1}^{N}\alpha_j\phi\left(\mathbf{x}^{(j)}\right)\right)$$
$$= \frac{1}{N}\sum_{i=1}^{N}\phi\left(\mathbf{x}^{(i)}\right)\left(\sum_{j=1}^{N}\alpha_j\phi\left(\mathbf{x}^{(i)}\right)^{\top}\phi\left(\mathbf{x}^{(j)}\right)\right)$$

mutliplying both cases terms by $\phi\left(\mathbf{x}^{(k)}\right)$

$$\lambda \sum_{j=1}^{N} \alpha_j \phi\left(\mathbf{x}^{(k)}\right)^\top \phi\left(\mathbf{x}^{(j)}\right) = \frac{1}{N} \sum_{i=1}^{N} \phi\left(\mathbf{x}^{(i)}\right) \phi\left(\mathbf{x}^{(k)}\right)^\top \left(\sum_{j=1}^{N} \alpha_j \phi\left(\mathbf{x}^{(i)}\right)^\top \phi\left(\mathbf{x}^{(j)}\right) \right)$$

$$\implies \lambda \sum_{j=1}^{N} \alpha_j G(k,j) = \frac{1}{N} \sum_{i=1}^{N} G(k,i) \sum_{j=1}^{N} \alpha_j G(i,j)$$

$$\implies \lambda (G\alpha)_k = \frac{1}{N} \sum_{j=1}^{N} \alpha_j (G^2)_{k,j}$$

$$\implies \lambda (G\alpha)_k = \frac{1}{N} (G^2\alpha)_k$$

$$\implies \lambda G\alpha = \frac{1}{N} G^2\alpha$$

$$\implies \lambda N\alpha = G\alpha.$$

with G being the Gram matrix. Links for the sources for this proof are also in this medium article and this paper

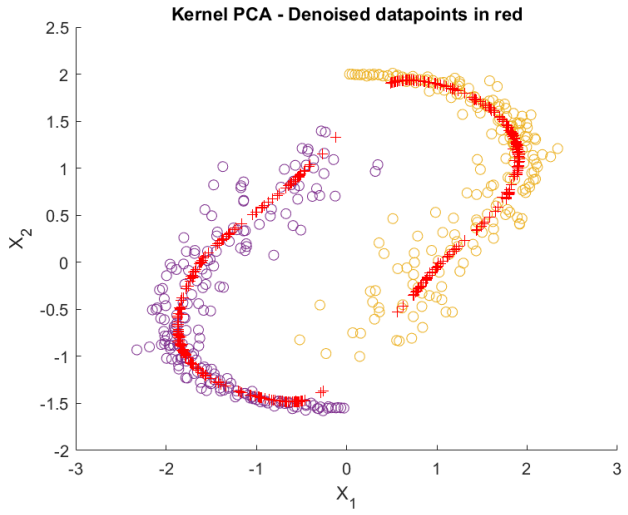

Figure 121: All Santa Fe data.

Q1.1.3: For the dataset at hand, propose a technique to tune the number of components, the hyperparameter and the kernel parameters.

Visualizing the Dataset in Figure 121, I think the best approach in non-linear kernels in general would be to use the RBF kernel, since it utilizes less parameters, namely only the sigma, and perform a grid search around a starting point that could be around $\frac{variance}{np}$ where n is the number of datapoints and p the input features, a relatively high C to ensure proper generalization and choosing a number of components that is around the elbow point in the high dimensional case, which should be too difficult to find since not a huge number of components would be utilized.

Q1.2.1: Explain briefly how spectral clustering works.

Spectral clustering utilizes the kernel matrix K used in the kernel PCA to create a Laplacian matrix $L = D - K$ where $D = \sum_{i=1}^{N} K_{ij}$ is the K column sum diagonal matrix. It then chooses the smallest eigenvalues, except the first since it is always 0 (the Laplacian is positive semi-definite and so all the eigenvalues are non-negative and 0 is always one of them) that correspond to the greatest clustering separation. Usually approximations are made to get the eigenvalues for the matrix K such as the normalized cuts algorithm or the random walk algorithm, utilized in our case, which takes the largest eigenvalues (except the first) of the random walk kernel matrix $P = D^{-1}A$.

It then projects the data in those dimensions and based on the projections, a clustering algorithm such as DBSCAN or k-means is utilized to perform clustering. In the case of binary clustering as in our case this is simply done using the sign of the resulting projections.

Q1.2.2: What are the differences between spectral clustering and classification?

Spectral clustering, as any type of clustering is unsupervised, it does not need to have the data labels to group data together. In contrast to that, classification is a supervised learning method, needing to have the class labels in order to perform the grouping and ascribe specific meaning to them.

Q1.2.3: Edit the script and try different values of sig2 (e.g., 0.001, 0.005, 0.01, 0.02). What is the influence of the sig2 parameter on the clustering results?

The sigma parameter influences the spread that is assumed the RBF kernel mapped data points have. The higher it is the wider the points are distributed and so even more remotely connected data points are deemed as grouped together (sig2=0.05 below). Decreasing it such as in values 0.02, 0.01, 0.005, 0.001 leads to perfect clustering since for all these values the projected points are only connected together with other points from their own group. In cases of even smaller sig2 values such as the case of 0.0001 many little islets of clusters are created, since the connectivity between projected points is very close, and close to 0. See below Figures 122-133.

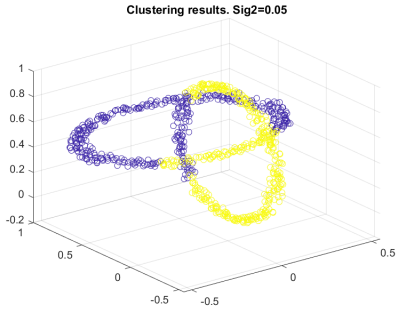

Figure 122

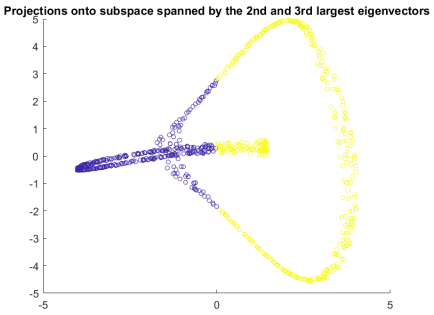

Figure 123

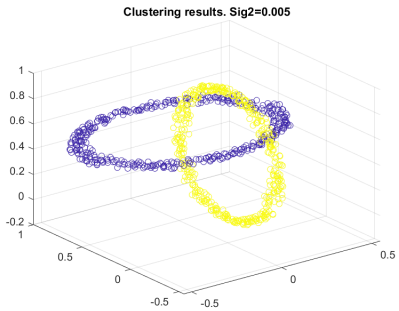

Figure 124

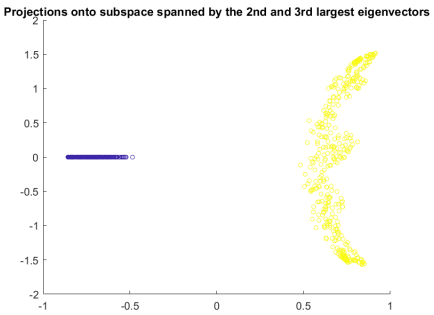

Figure 125

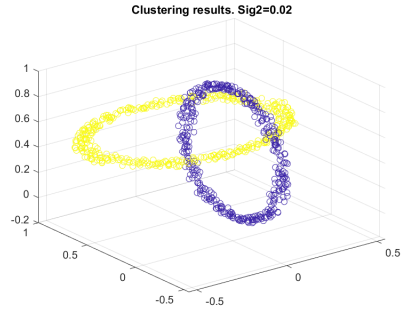

Figure 126

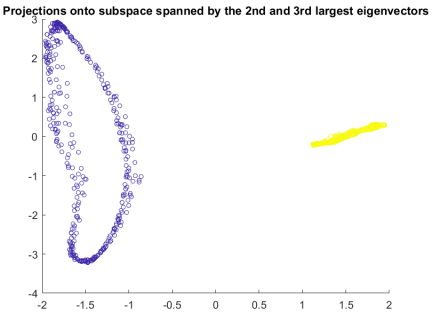

Figure 127

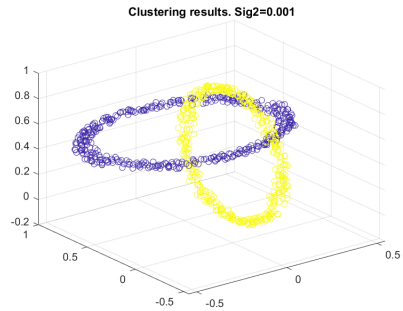

Figure 128

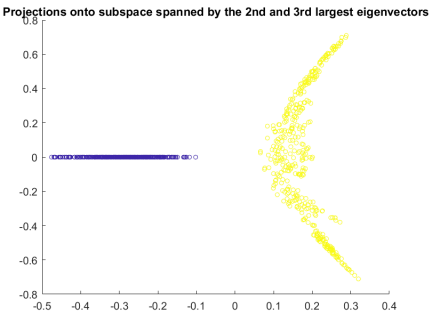

Figure 129

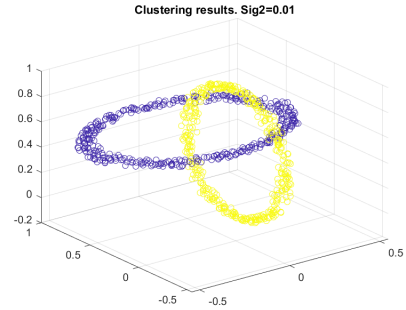

Figure 130

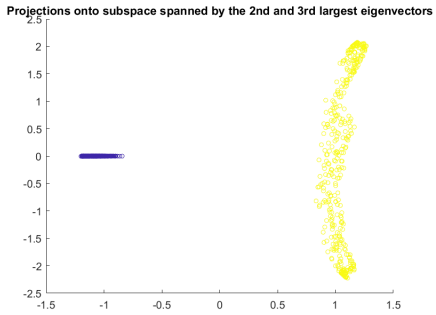

Figure 131

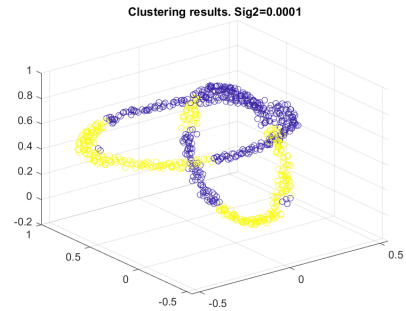

Figure 132

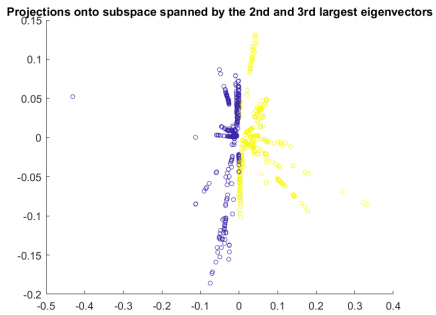

Figure 133

Q1.3.1: In which setting would one be interested in solving a model in the primal? In which cases is a solution in the dual more advantageous?

There are many reasons I think for the use of the dual form. The first has to do with the curse of dimensionality. The linear SVM that utilizes the primal form scales as $\mathcal{O}(np)$ which is very inefficient for high dimensions, even much more so when trying to applying the kernel trick, increasing them, however the dual form scales with $\mathcal{O}(n^2)$, thus for high dimensionalities it is highly efficient, enabling the use of kernels. Stemming from that, in case of linearly data a linear SVM can suffice but given that a lot of datasets adhere to nonlinear distributions, the dual form is needed to be able to apply a kernel mapping.

Q1.3.2: What is the effect of the chosen kernel parameter sig2 on the resulting fixed-size subset of data points (see fixedsize script1.m)? Can you intuitively describe to what subset the algorithm converges?

Methods like the Nystrom or the random walk are utilized in order to reduce the complexity of kernel algorithms without sacrificing much in aspects of performance. In order to retain the most information, and pick certain fixed points for the kernel representation, the entropy is maximized. The kernel's hyperparameters, however plays a huge role again. Given a small σ^2 the points picked could be very close to each other since the kernel will regard them as being far away, conversely a large kernel will widen the search radius of a picked point, increasing the probability that more outlying points are going to be picked. The results for varying σ^2 are displayed below in Figures 134-139, where we can see the aforementioned. I think the best results would be σ^2 with a midrange value so that edge cases are picked as well as points in denser regions so that the density of the picked data points resembles the original.

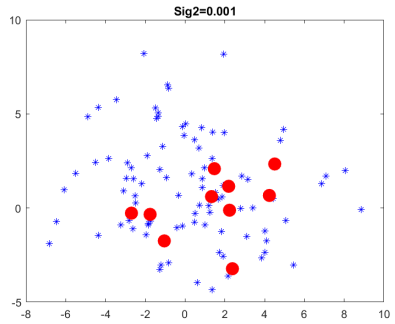

Figure 134

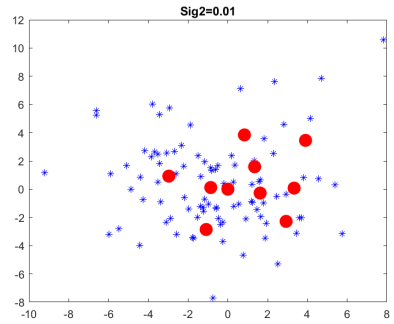

Figure 136

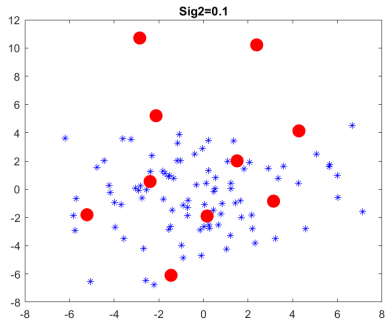

Figure 138

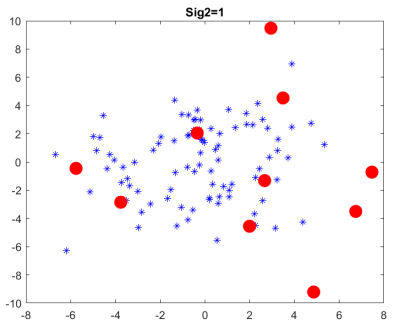

Figure 135

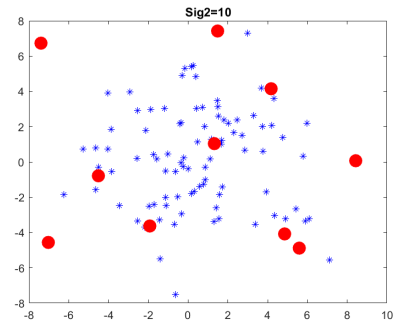

Figure 137

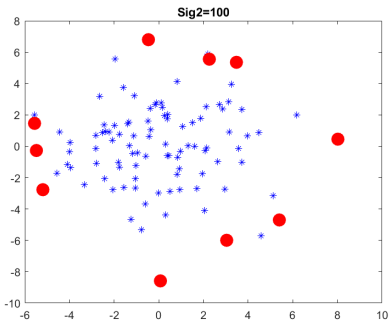

Figure 139

Q1.3.3: Run fslssvm_script.m. Compare the results of fixed-size LS-SVM to l0-approximation in terms of test errors, number of support vectors and computational time.

The l0-approximation tries to create sparser solutions by adding a penalty in the cost function that is proportional to the number of zeros a weight vector has. This is aimed at creating sparser models, with less support vectors and thus memory requirements, however is not used regularly being discontinuous and non-convex making the optimization difficult to manage whereas the l1 norm is preferred.

Below the results are displayed in Figures 140-145. It can be easily seen, that the time taken is higher when applying the l0-approximation, which is to be expected given that the l0-approximation occurs after the fixed-size LS-SVM. Next, when we compare the other 4 figures we see something extremely interesting. First, given less support vectors, we see that the error increases, which is to be expected since the model has less granularity in the trade-off for memory efficiency. Next, when having the same number of support vectors, the errors are not the same, but what I find baffling is that the cost function seems to be slightly better in the case of the l0 approximation, even if by a small amount, even though one would expect it to be higher..

Q2.1.1: Illustrate the difference between linear and kernel PCA by giving an example of digit denoising for noisefactor = 1.0. Give your comments on the results (based on visual inspection)

In figures 146 and 147 are the kernel and linear PCA respectively. Given that that the noise is not linearly separable we can see that as the number of components increase the linear PCA adds that uniform noise back into the images, while kernel PCA seems to be able to ignore the noise given the kernel mapping that can handle non-linearities. Also , given enough components the kernel PCA seems to be able to recosntruct digits very clearly, however at the cost of accuracy, reconstructing the wrong digits. This is due to the non-linear separation of the noise that takes into it information represented bu the original digits but seriously distorted by the noise.

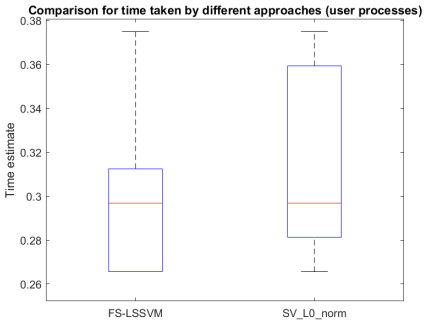

Figure 140

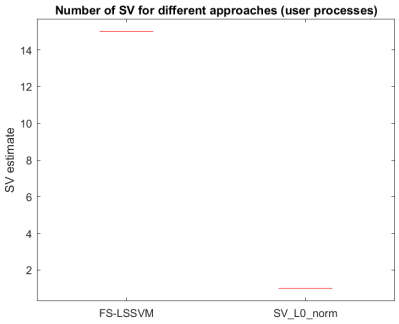

Figure 142

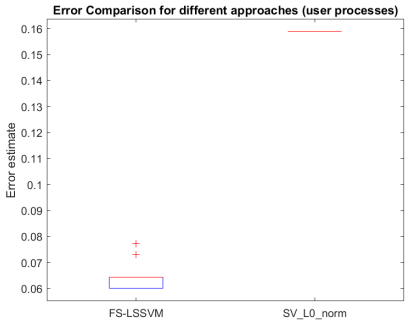

Figure 144

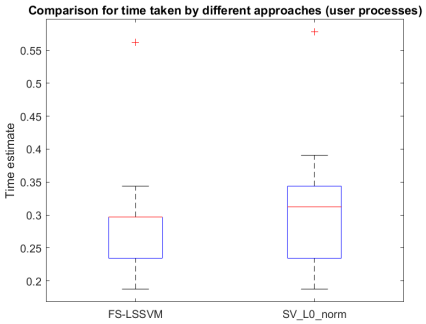

Figure 141

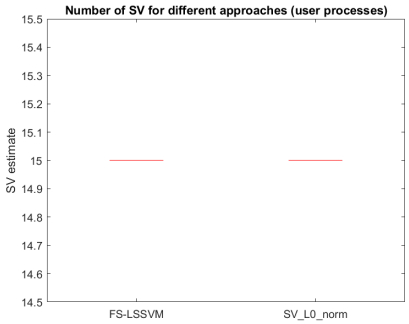

Figure 143

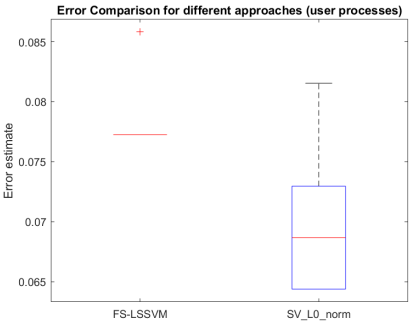

Figure 145

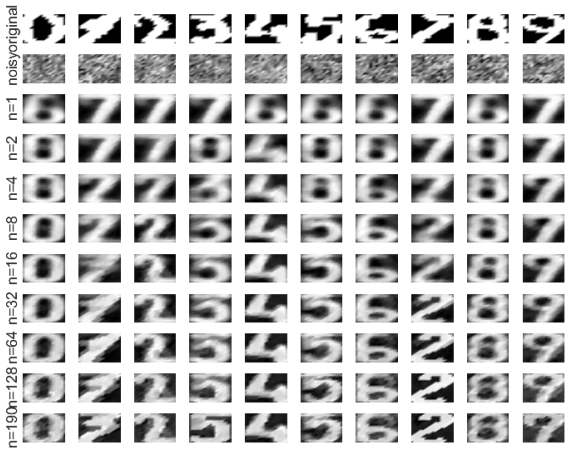

Figure 146: Kernel PCA

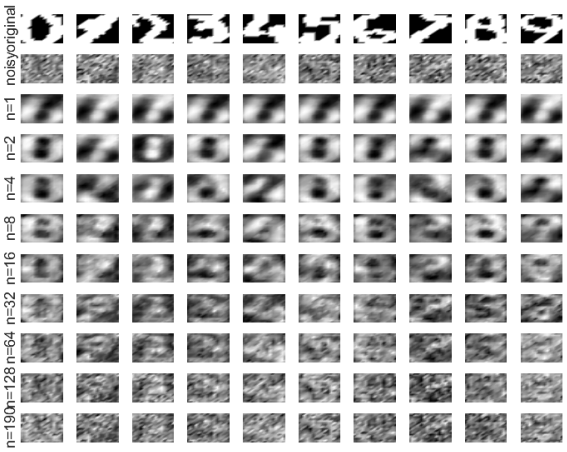

Figure 147: Linear PCA

Q2.1.2: What happens when the sig2 parameter is much bigger than the suggested estimate? What if the parameter value is much smaller? In order to investigate this, change the sigma factor parameter for equispaced values in logarithmic scale.

When changing the parameter sigmafactor (or σ^2, since it is a scaling factor of it) we can increase or decrease the radius of variance each datapoint is considered to have. When the sigma is too large, the resulting mapping function is very smooth and as in Figure 148 below we can see that the noise cannot be effectively reduced apart from the initial principal components, which however are inaccurately reconstructed, and after enough components, only noise remains. That is due to the effective radius that groups together points of both noise and of the original digit information. As the σ continues to decrease we reach an optimal point such as Figure 152, where the reconstruction has a significant amount of noise on the one hand, but is very accurate. Further decreasing σ leads to clearer but less accurate images, since the reduced effective radius can separate noise from information, but overfits probably on irrelevant training data details due to the actual digits that are closer together in the input space. Decreasing the σ even more leads to practically every point being considered too distant from every other one, as in Figure 153 and thus the noise cannot be separated from relevant digit information.

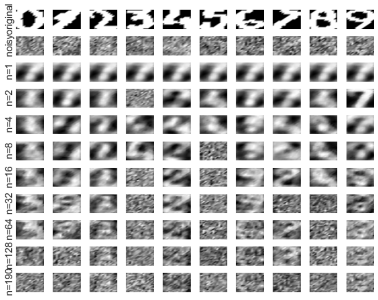

Figure 148: sigmafactor=100

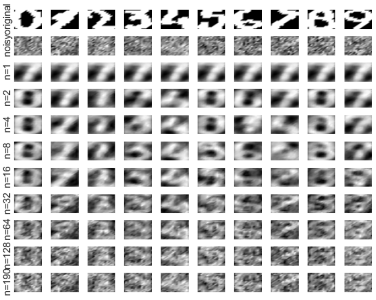

Figure 150: sigmafactor=10

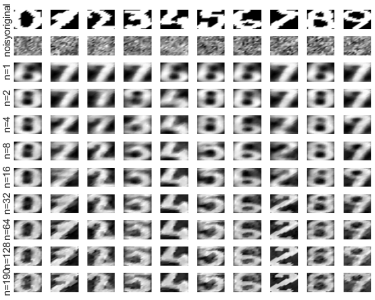

Figure 152: sigmafactor=1

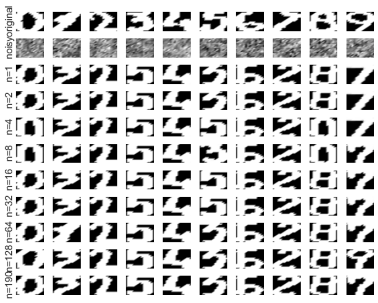

Figure 149: sigmafactor=0.1

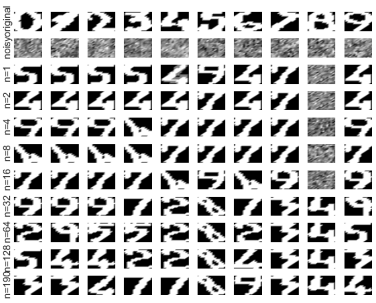

Figure 151: sigmafactor=0.01

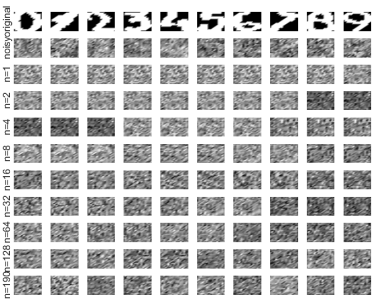

Figure 153: sigmafactor=0.001

Q2.1.3: Investigate the reconstruction error on training (Xtest) and validation sets (Xtest1 and Xtest2), as a function of the kernel PCA denoising parameters. Select parameter values such that the error on the validation sets is minimal. Can you observe any improvements in denoising using these optimized parameter settings?

It is obvious from the previous results that the best scores were in the 190 principal vector space, so I kept that and then performed a grid search around sigmafactor 1 to find the minimal sum of MSE error betweem Test1 and Test2, which occured for a sigmafactor of 0.8. In figures 154-156 I present the reconstructed images. Their quality is beyond what I could imagine one would hope to achieve while the total MSE error for the 20 digits for the 2 test sets was 1.309e+4.

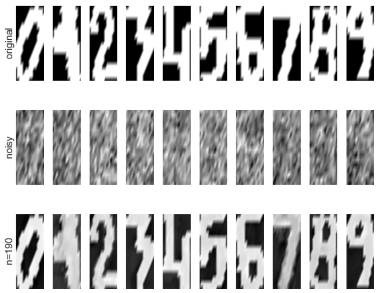

Figure 154: Traininig Set, sigmafactor=0.8

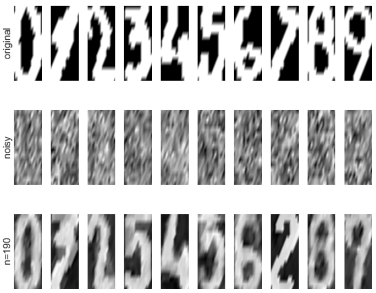

Figure 155: Test Set 1, sigmafactor=0.8

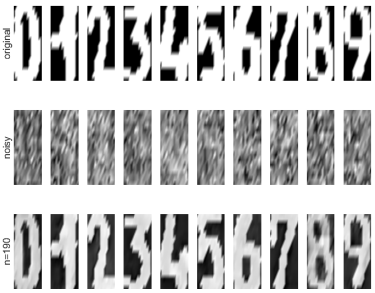

Figure 156: Test Set 2, sigmafactor=0.8

Q2.2.1: Shuffle Dataset - Explore and visualize (part of) the dataset. How many datapoints? How many and meaning of attributes? How many classes? What is to be expected about classification performance? Visualize the results.

The shuttle dataset consists of 58000 datapoint with 9 attributes, one being the time and the other 8 being attributes of radiators concerning space programmes. The attributes are time, Rad Flow, Fpv close, Fpv open, High, Bypass, Bpv close, Bpv open. The last tenth column contains the 7 classes, the 1 of which is the majority with around 80% of the total cases. Thus the classification performance of an accuracy task would be 80% for the random model while for the ROC the AUC should be above 0.8. For tasks like these, I prefer the averaged F1 score or weighted accuracy to mitigate exactly these issues. We will perform the analysis on 1000 datapoints only, splitting them 70% for training and 30% for testing. First let's visualize the attributes of all datapoints in Figures 157-165.

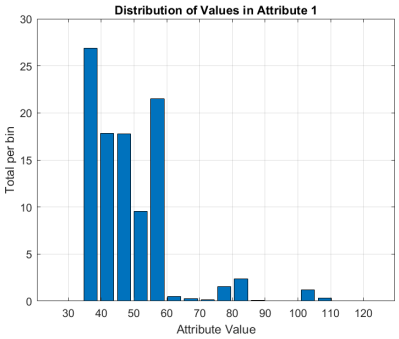

Figure 157

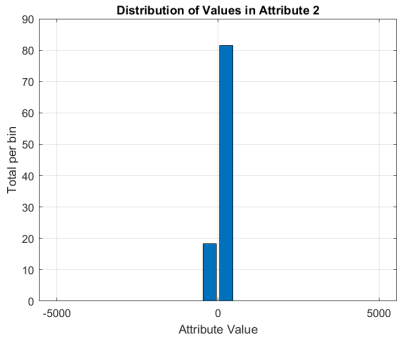

Figure 160

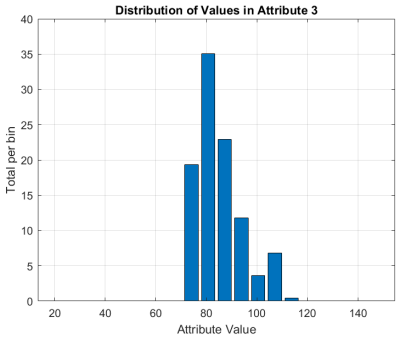

Figure 163

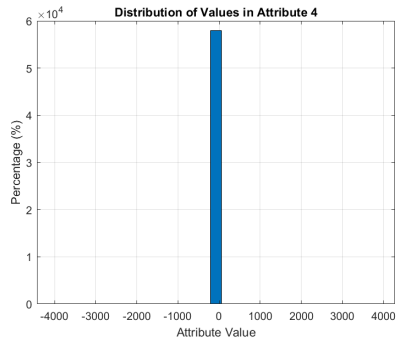

Figure 158

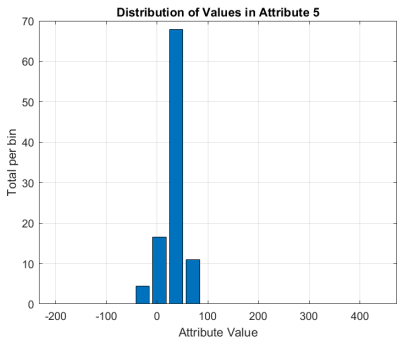

Figure 161

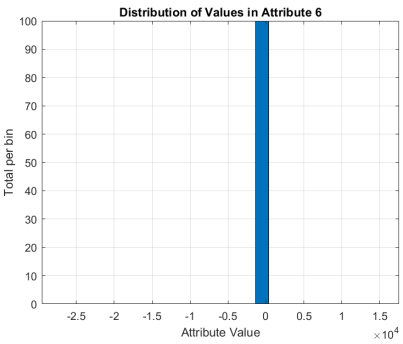

Figure 164

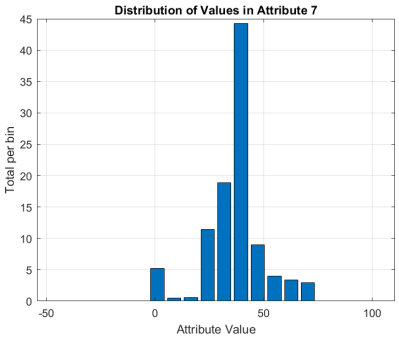

Figure 159

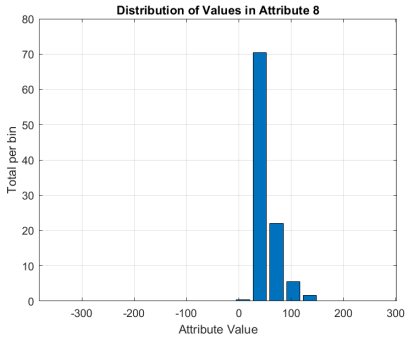

Figure 162

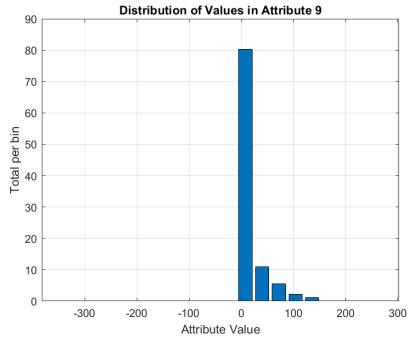

Figure 165

Attribute 1 is time and is not uniform so the data is not structured as a time series. Furthermore, attributes 1 and 3 contain the most variance and thus are best for visualization. Owing to that I will showcase the superiority of the kernel PCA here showing that while the linear PCA needs 5 principal components to relatively accurately recreate the data, kernel PCA needs only one (Figures 166-168). The test set comprised of 300 more data points can be seen in Figure 169.

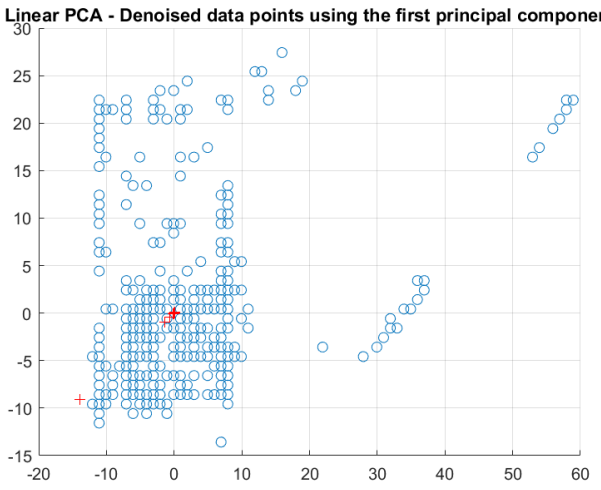

Figure 166: Linear PCA
1 component

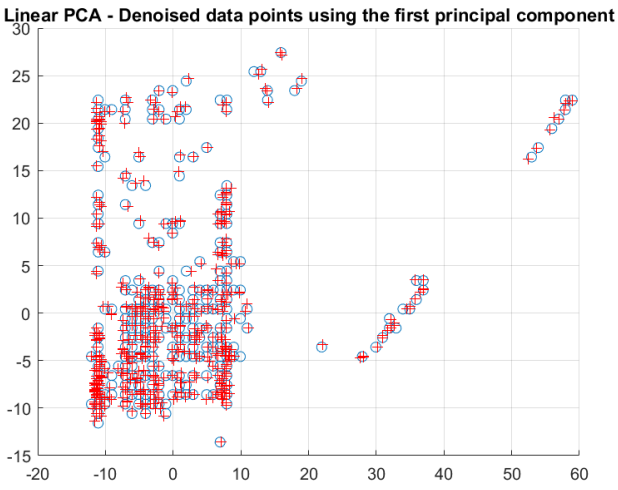

Figure 167: Linear PCA (train)
5 - components

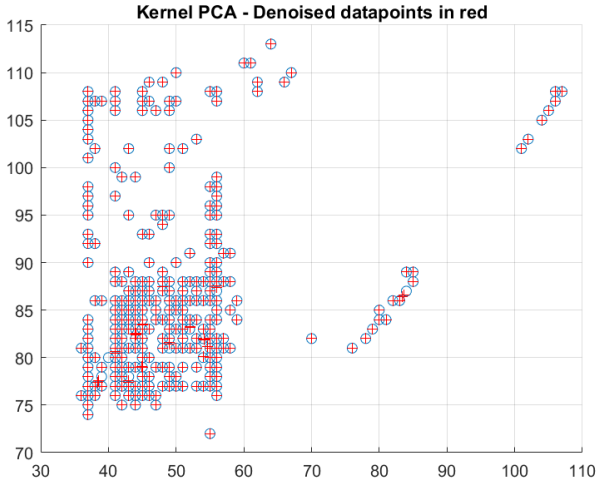

Figure 168: Kernel PCA (train)
1 component

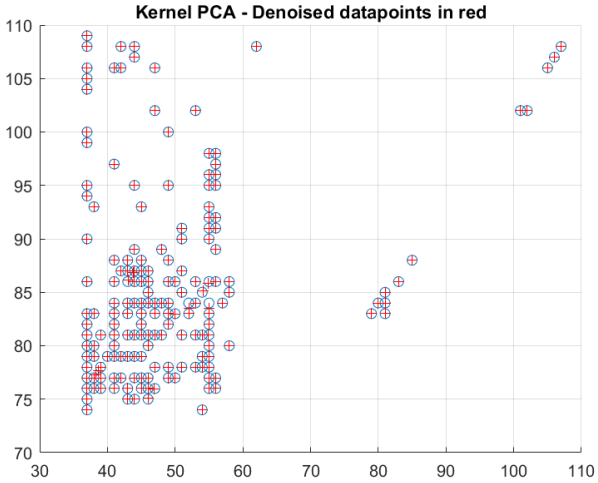

Figure 169: Kernel PCA (test)
1 component

The results I got with the fixed size LS-SVM are displayed below in Figures 170-172. Where we see that the l0 approximation takes more time and reaches greater error for the same number of support vectors.

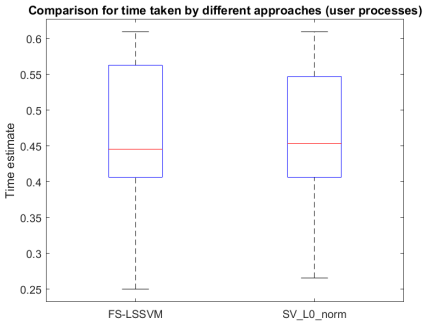

Figure 170

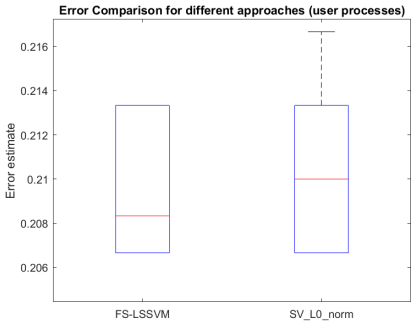

Figure 171

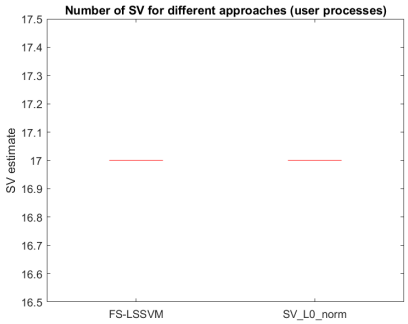

Figure 172

Q2.2.2: California Dataset - Explore and visualize (part of) the dataset. How many datapoints? How many and meaning of attributes? Visualize and explain the obtained results?

The california dataset contains information about the median house value of blocks in California based on the 1990 census data. The dataset contains 20640 datapoints with 8 variables: longitude, latitude, housingMedianAge, totalRooms, totalBedrooms, population, households, medianIncome from which we are tasked to predict the ninth dependent variable which is the medianHouseValue. In the same manner as before, I visualize the data to get a first peak into the dataset.

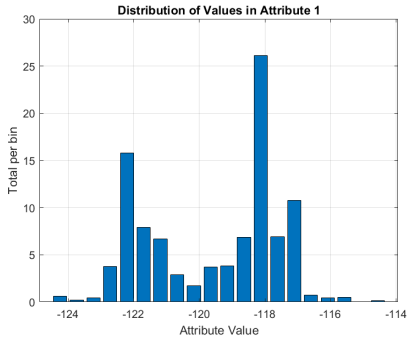

Figure 173

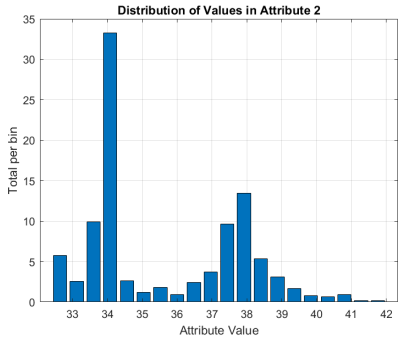

Figure 174

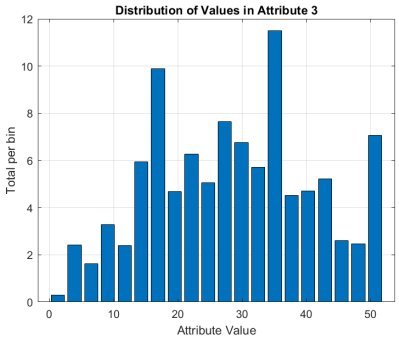

Figure 175

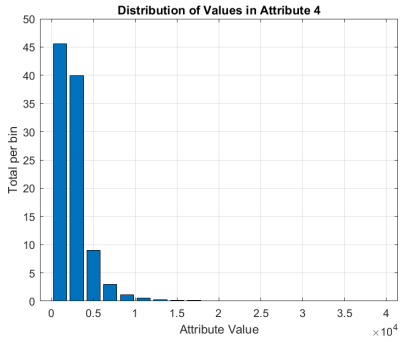

Figure 176

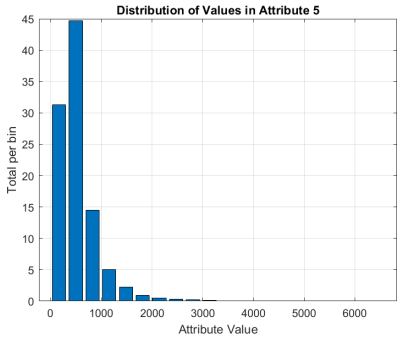

Figure 177

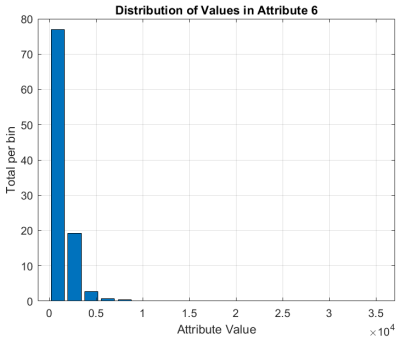

Figure 178

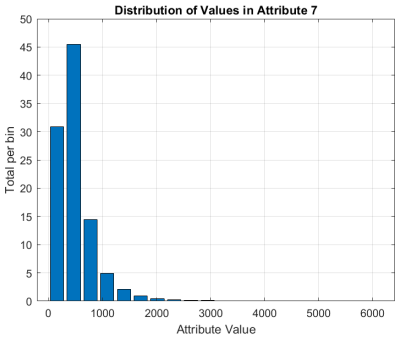

Figure 179

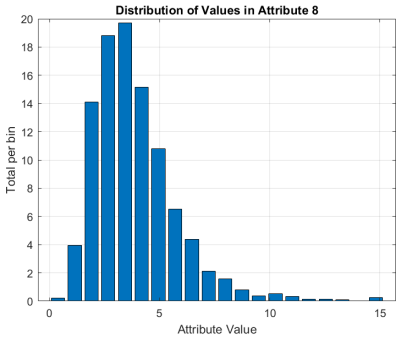

Figure 180

Performing the same test for the fixed size LS-SVM splitting the dataset into 75% training and 25% test data I got the below results in Figures 181-183. Here the l0 approximation gives better results decreasing by a very large amount the number of support vectors, achieving higher sparseness without sacrificing time or perfromance. Hence, from the two datasets a hypothesis could be made that the l0 approximation is better the larger the dataset is.

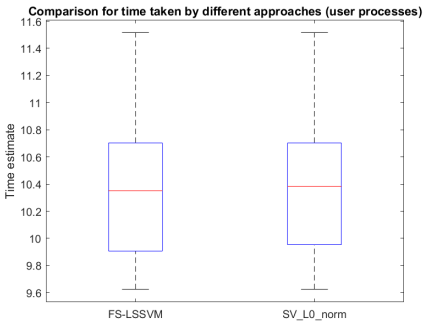

Figure 181

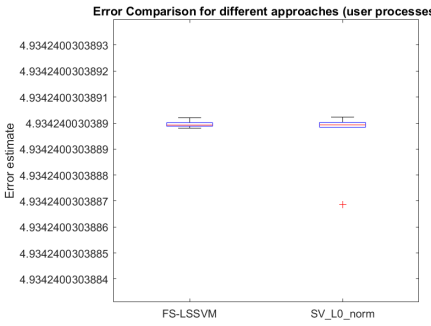

Figure 182

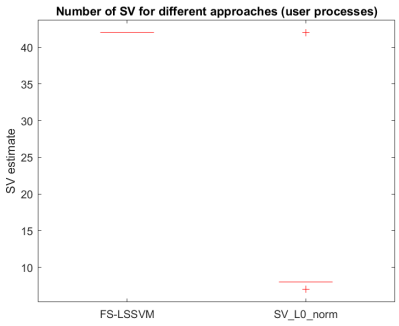

Figure 183

Use of ChatGPT (or any other AI writing assistance tool)

Form to be completed

Student name: Dimitrios Chatzakis

Student number: r0977164

Please indicate with "X" whether it relates to a course assignment or to the master thesis:

O This form is related to a **course assignment**.

Course name: Support Vector Machines: Methods and Applications

Course number: H

O This form is related to **my Master thesis**.

Title Master thesis: .....

Promotor: .....

Please indicate with "X":

O **I did not use** ChatGPT or any other AI writing assistance tool.

O **I did use** AI Writing Assistance. In this case **specify which one** (e.g. ChatGPT/GPT4/...):

ChatGPT 4o .....

Please indicate with "X" (possibly multiple times) in which way you were using it:

O **Assistance purely with the language of the paper**

- *Code of conduct*: This use is similar to using a spelling checker

O **As a search engine to learn on a particular topic X**

- *Code of conduct*: This use is similar to e.g. a google search or checking Wikipedia. Be aware that the output of Chatbot evolves and may change over time.

O **For literature search X**

- *Code of conduct*: This use is comparable to e.g. a google scholar search. However, be aware that some AI writing assistance tools like ChatGPT may output no or wrong references. As a student you are responsible for further checking and verifying the absence or correctness of references.

O **For short-form input assistance**

- *Code of conduct*: This use is similar to e.g. google docs powered by generative language models

O **To let generate programming code X**

- *Code of conduct*: Correctly mention the use of ChatGPT (or other AI writing assistance tool) and cite it. You can also ask ChatGPT how to cite it.

O **To let generate new research ideas**

- *Code of conduct*: Further verify in this case whether the idea is novel or not. It is likely that it is related to existing work, which should be referenced then.

O **To let generate blocks of text**

- *Code of conduct*: Inserting blocks of text without quotes from ChatGPT (or other AI writing assistance tool) to your report or thesis is not allowed. According to Article 84 of the exam regulations in evaluating your work one should be able to correctly judge on your own knowledge. In case it is really needed to insert a block of text from ChatGPT (or other AI writing assistance tool), mention it as a citation by using quotes. But this should be kept to an absolute minimum.

O **Other**

- *Code of conduct*: Contact the professor of the course or the promotor of the thesis. Inform also the program director. Motivate how you comply with Article 84 of the exam regulations. Explain the use and the added value of ChatGPT or other AI tool: ….

Further important guidelines and remarks

- ChatGPT cannot be used related **to data or subjects under NDA agreement.**
- ChatGPT cannot be used related **to sensitive or personal data due to privacy issues**.
- **Take a scientific and critical attitude** when interacting with ChatGPT (or other AI writing assistance tool)and interpreting its output. Don't become emotionally connected to AI tools.
- As a student you are responsible to comply with Article 84 of the exam regulations: your report or thesis should reflect your own knowledge. Be aware that plagiarism rules also apply to the use of ChatGPT or any other AI tools.
- **Exam regulations Article 84**: "Every conduct individual students display with which they (partially) inhibit or attempt to inhibit a correct judgement of their own knowledge, understanding and/or skills or those of other students, is considered an irregularity which may result in a suitable penalty. A special type of irregularity is plagiarism, i.e. copying the work (ideas, texts, structures, designs, images, plans, codes , ...) of others or prior personal work in an exact or slightly modified way without adequately acknowledging the sources. Every possession of prohibited resources during an examination (see article 65) is considered an irregularity."
- **ChatGPT suggestion about citation**: "Citing and referencing ChatGPT output is essential to maintain academic integrity and avoid plagiarism. Here are some guidelines on how to correctly cite and reference ChatGPT in your Master's thesis: 1. Citing ChatGPT: Whenever you use a direct quote or paraphrase from ChatGPT, you should include an in-text citation that indicates the source. For example: (ChatGPT, 2023). 2. Referencing ChatGPT: In the reference list at the end of your thesis, you should include a full citation for ChatGPT. This should include the title of the AI language model, the year it was published or trained, the name of the institution or organization that developed it, and the URL or DOI (if available). For example: OpenAI. (2021). GPT-3 Language Model. https://openai.com/blog/gpt-3-apps/ 3. Describing the use of ChatGPT: You may also want to describe how you used ChatGPT in your research methodology section. This could include details on how you accessed ChatGPT, the specific parameters you used, and any other relevant information related to your use of the AI language model. Remember, it is important to adhere to your institution's specific guidelines for citing and referencing sources in your Master's thesis. If you are unsure about how to correctly cite and reference ChatGPT or any other source, consult with your thesis advisor or a librarian for guidance."

Additional reading

ACL 2023 Policy on AI Writing Assistance: https://2023.aclweb.org/blog/ACL-2023-policy/

KU Leuven guidelines on citing and referencing Generative AI tools, and other information:

https://www.kuleuven.be/english/education/student/educational-tools/generative-artificial-intelligence

Dit formulier werd opgesteld voor studenten in de Master of Artificial intelligence. Ze bevat een code of conduct, die we bij universiteitsbrede communicatie rond onderwijs verder wensen te hanteren.

Deze code of conduct schept een kader voor academiejaar 2023-2024, aanpassingen kunnen gebeuren in functie van nieuwe evoluties.