Support Vector Machine Assignments

[H00H3a]

Advanced Master in Artificial Intelligence

Option Engineering and Computer Science

Supervisor: Prof. Johan Suykens

Student name: Zhiqi Wang

Student ID: r0822618

August, 2023

1. Assignment 1: Classification

1.1 A simple example: two Gaussians

The dataset consists of normally distributed random numbers generated using the *randn* function. It comprises two classes generated from Gaussian distributions with centers at (1, 1) and (-1, -1). These classes share the same covariance matrix. Thus, a linear classifier passing through the origin (0, 0) aims to maximize the separation between the class means and the classifier. Figure 1 illustrates the optimal line as y = -x + 4, which minimizes misclassification despite the data not being perfectly linearly separable.

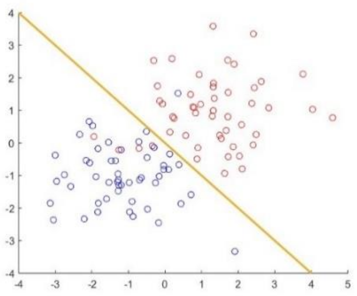
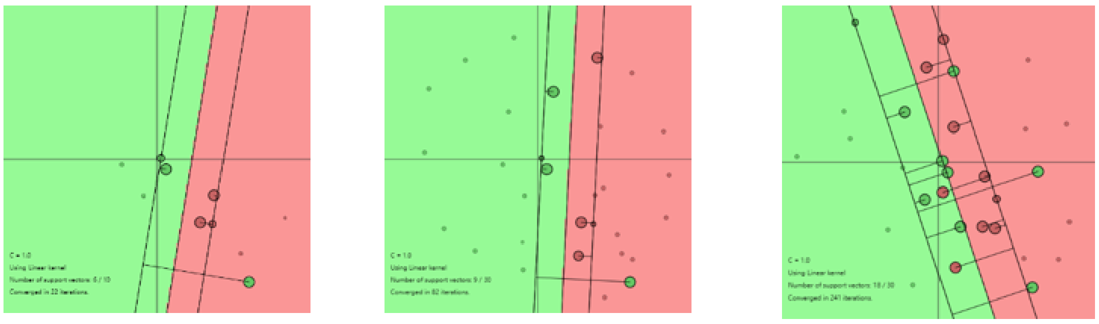


Figure 1 - Linear classification

1.2 Support vector machine classifier

- ***Add more data point***

Support vectors are defined as data points in proximity to the hyperplane, playing a role in determining the placement and orientation of said hyperplane. Data points positioned near the decision boundary are inclined to be identified as support vectors. The initial state, depicted in the left image of Figure 2, illustrates that data points in closer proximity to the hyperplane are visually magnified. This magnification underscores their significance in the classification process. Consequently, the hyperplane's position can be ascertained through the maximization of the margin.

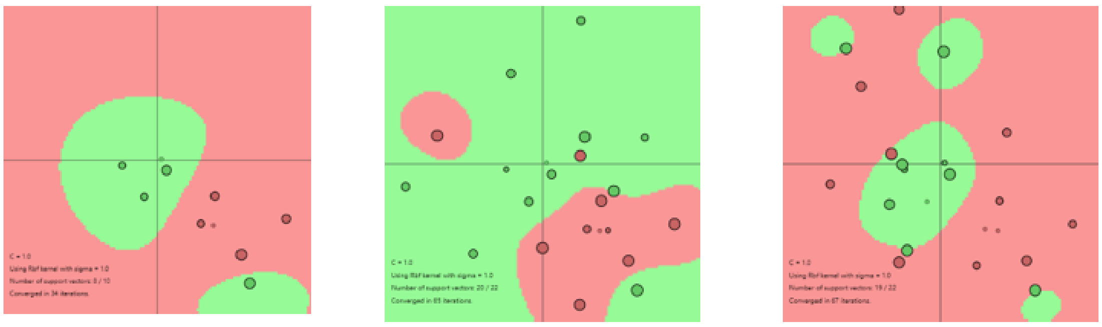


```
C = 1.0
Using Linear kernel
Number of support vectors: 6 / 10
Converged in 22 iterations.
```

```
C = 1.0
Using Linear kernel
Number of support vectors: 9 / 30
Converged in 82 iterations.
```

```
C = 1.0
Using Linear kernel
Number of support vectors: 18 / 30
Converged in 241 iterations.
```

Figure 2 - SVM classifier with linear kernel. The original data point (left). Add more data points on the right sides (Middle). Add more data points on the wrong side (right):



```
C = 1.0
Using RBF kernel with sigma = 1.0
Number of support vectors: 8 / 10
Converged in 34 iterations.
```

```
C = 1.0
Using RBF kernel with sigma = 1.0
Number of support vectors: 20 / 22
Converged in 85 iterations.
```

```
C = 1.0
Using RBF kernel with sigma = 1.0
Number of support vectors: 19 / 22
Converged in 67 iterations.
```

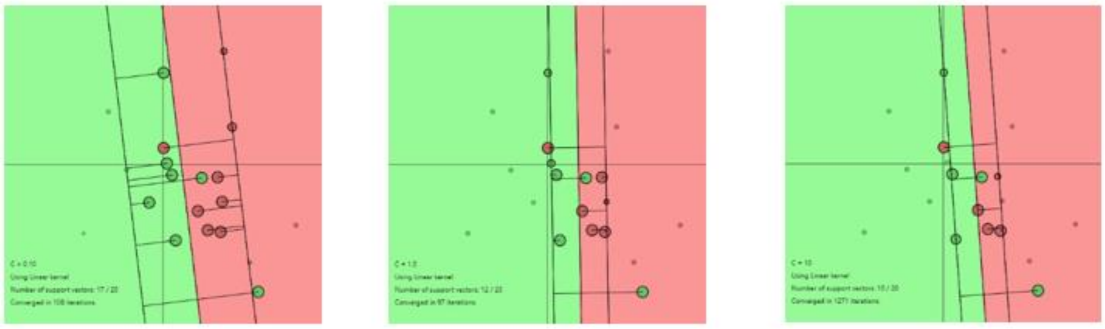
Figure 3 - The SVM classification with RBF kernel. The original data point (Left). Add more data points on the right sides (Middle). Add more data points on the wrong side (Right)

As we introduce additional points, adjustments to the classification boundary occur only when these new data points influence the existing conditions. If the newly added points are distant from the hyperplane and are correctly classified, the hyperplane remains unaltered.

The middle and right images of Figure 2 exemplify this behavior. For the linear kernel, adding more data points to the right side results only leads to minimal modifications to the hyperplane, its margins, and the count of support vectors. However, a noteworthy shift transpires if data points are added on the wrong side.

For the RBF kernel, as shown in Figure 3, the introduction of additional data points on the right side serves to enlarge the region associated with the class to which the new points belong. Conversely, when more points are added on the wrong side, the initial region becomes fragmented into smaller, isolated segments. This behavior signifies the profound impact that the placement of data points can have on SVM classification.

- ***Try different regularization hyperparameter C***



```
C = 0.10
Using Linear kernel
Number of support vectors: 17 / 20
Converged in 108 iterations.
```

```
C = 1.0
Using Linear kernel
Number of support vectors: 12 / 20
Converged in 97 iterations.
```

```
C = 10
Using Linear kernel
Number of support vectors: 10 / 20
Converged in 1271 iterations.
```

Figure 4 - The SVM classifier with linear kernel for different choices of C parameter. (Left to right :0.1, 1, 10).

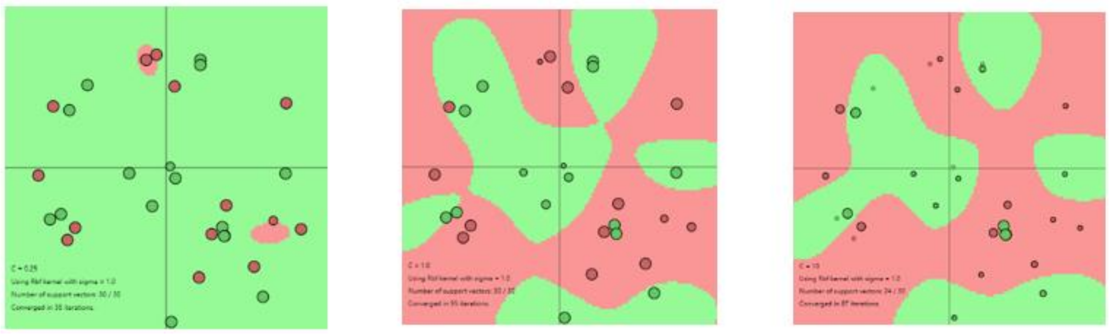


Figure 5 - The SVM classifier with RBF kernel for different choices of C parameter. (Left to right: 0.25, 1, 10).

In addition to the positioning of added data points, the C parameter wields influence over the significance of support vectors. This parameter serves as a regularization factor, balancing the cost of misclassification against the margin of the decision function. A higher value of C translates to a diminished tolerance for errors, prioritizing accurate classification of data points and subsequently leading to a narrower margin.

A discernible pattern emerges from Figure 4 concerning linear kernels. As the C parameter's value escalates, the margin contracts, and the count of considered support vectors increases. The impact of the RBF kernel, on the other hand, can be gleaned from Figure 5. When the C value experiences an upsurge, the threshold for permitting misclassifications diminishes, fostering a smoother and more refined decision boundary.

- ***Try different kernel parameter sigma***

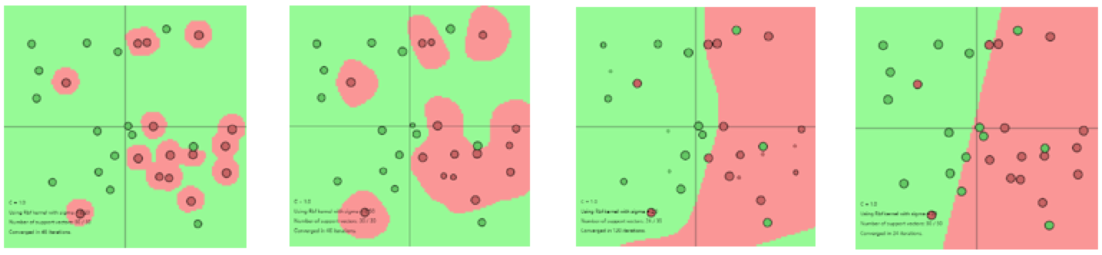


Figure 6 - The SVM classifier with RBF kernel for different choices of σ^2 value.

The kernel coefficient σ defines the influence range of a data point. Higher σ values group more distant data points into the same class, leading to linear decision boundaries. Lower σ values require closer data points to belong to the same class, resulting in highly nonlinear boundaries and a risk of overfitting.

Figure 6 highlights this behavior: excessively small or large σ values both lead to overfitting. Smaller σ values create complex, isolated boundaries, while larger σ values cause the model to miss capturing the data distribution. This emphasizes the crucial role of parameter tuning in SVMs.

- ***Linear kernel vs RBF kernel***

The linear kernel excels when data separation is clear and classes are linearly separable, offering computational efficiency. However, it struggles with strong nonlinear data. In contrast, the RBF kernel is adaptable and captures complex relationships, but demands more computation and risks overfitting. Correct parameter tuning is crucial for both kernels, as linear kernels may underperform with nonlinear data, while RBF kernels can falter if parameters are chosen incorrectly. The choice depends on data complexity, demanding careful consideration.

1.3 Least-squares support vector machine classifier

1.3.1 Influence of hyperparameters and kernel parameters

To simplify SVM's complex formula, we introduce the LS-SVM technique, which relaxes SVM's constraints. An LS-SVM classifier is trained on the Iris dataset, featuring 100 training samples and 20 test samples in a two-dimensional, two-class layout. Employing a polynomial kernel on the test set, we explore the effect of various degrees on classification and compute the misclassification rate.

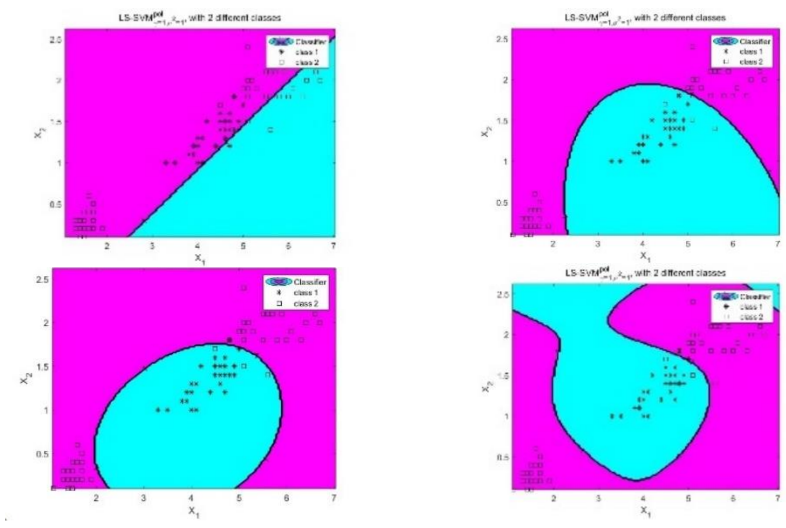


Figure 7 - The polynomial kernel with degrees (Left to right, top to bottom: Degree =1, error =55%; Degree =2, error rate =5%; Degree =3, error rate =0; Degree =4, error rate =0)

In Figure 7, an intriguing trend emerges: with increasing degree, the misclassification error diminishes. When the degree equals 1, the impact mirrors that of a linear kernel. Polynomial degrees of 3 adeptly capture the data distribution, striking a balance between accuracy and complexity. However, starting from a degree of 4, the polynomial's complexity becomes excessive, possibly leading to overfitting. This observation underscores the importance of optimal parameter selection for effective LS-SVM classification.

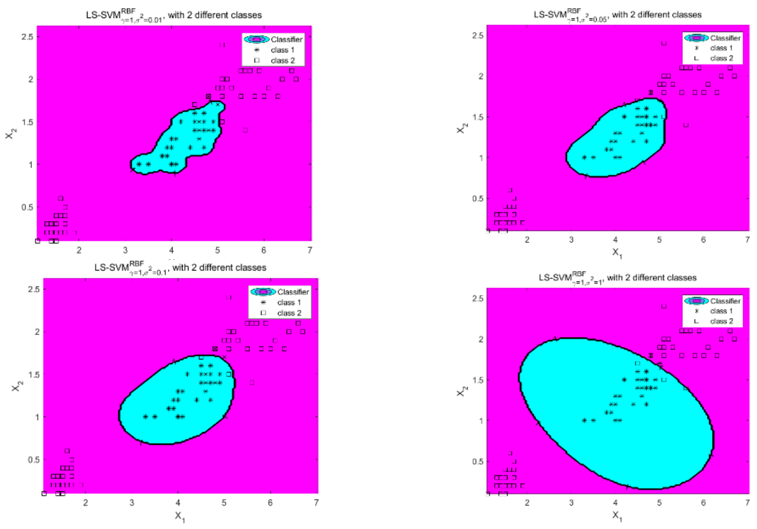


Figure 8 - The polynomial kernel with different σ^2 (Left to right, top to bottom: $\sigma^2 = 0.01, 0.05, 0.1, 1$) with fix $\gamma = 1$

Figure 8 demonstrates RBF kernel classification, with a fixed Gamma (γ) of 1 and varied σ^2 values. Increasing σ^2 leads to a smoother decision boundary and zero classification error. Extremely low σ^2 (e.g., 0.01) makes the model overly complex, while $\sigma^2 = 1$ underfits and $\sigma^2 = 0.1$ strikes a favorable balance between complexity and fitting. Precise parameter selection proves crucial when utilizing the RBF kernel for classification, as shown in this analysis.

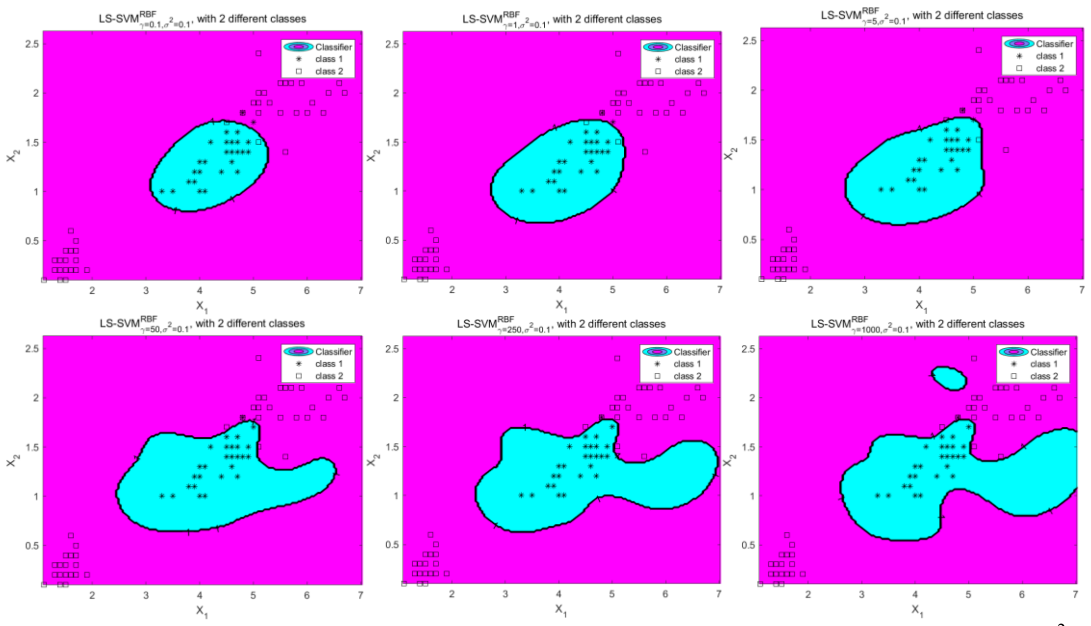


Figure 9 - Tune the different γ (Left to right, top to bottom: $\gamma = 0.1, 1, 10, 50, 250, 1000$) with fix $\sigma^2 = 0.1$

Gamma (γ) is a regularization parameter that balances fitting error and function smoothness. Figure 9 shows RBF kernel classification with $\sigma^2 = 1$ and varying γ values. As γ increases, classification error decreases. $\gamma = 10$ or 50 fit well, but excessive γ (e.g., $\gamma = 1000$) causes misclassification. Optimal γ range is about 1-10. Similarly, from Figure 10, ideal σ^2 range is around 0.01-10. Effective parameter tuning ensures accurate RBF kernel performance.

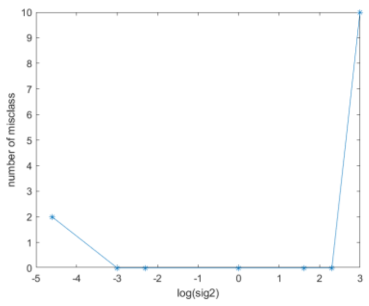


Figure 10 - The misclassification of different value of σ^2.

1.3.2 Tuning parameters using validation.

To assess the three validation methods, a set of 6 gamma and 6 sigma values were chosen for combined testing, resulting in heatmaps for each approach. In these heatmaps, lighter shades indicate higher accuracy and lower misclassification rates. Figure 11 displays the outcomes.

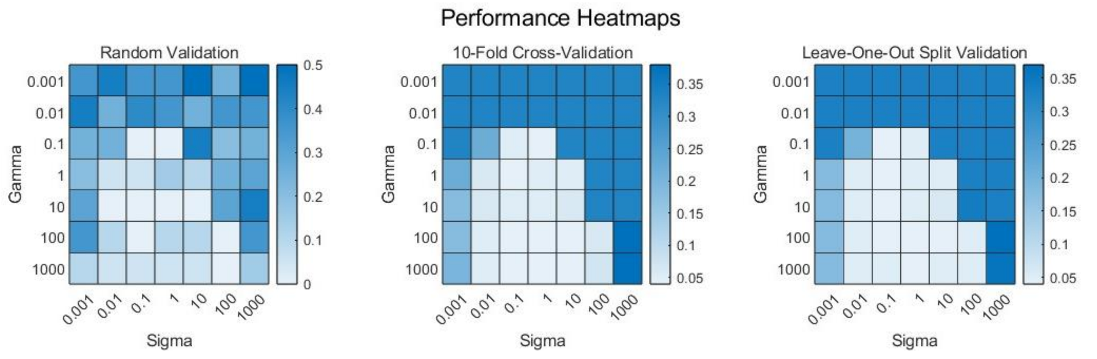


Figure 11 - The performance of each validation methods (left to right: Random split, k-fold, leave-one-out). The color represents accuracy.

Comparing the methods, cross-validation emerges as more pessimistic than random splitting. Cross-validation yields smoother transitions in color changes, offering a continuous assessment. In contrast, random splitting showcases disjointed color variations among cells. Remarkably, leave-one-out closely resembles cross-validation, with peak accuracy concentrated between gamma=1-1000 and sig2=0.01-10. Notably, across multiple runs, cross-validation exhibits the most consistent results.

Random split validation involves randomly dividing the dataset into training and validation sets. However, this method can yield inflated accuracy that doesn't truly represent classifier performance. To address this, we use cross-validation. It segments the dataset into k folds, training on k-1 and evaluating on the left-out fold. K-fold cross-validation calculates average precision across iterations, enhancing robustness, but it demands computational resources. For smaller training sets, a variation is k-fold cross-validation with k equal to the number of samples. Here, only one sample is left out per iteration for training. The random validation is slightly more optimistic than traditional cross-validation.

The selection of the value "k" in cross-validation warrants attention. Optimal values typically fall between 5 and 10. A value too large diminishes sample combinations, resembling random splitting, while also increasing computational demands. This stems from the fewer iterations possible due to limited sample diversity. Therefore, a fragile balance must be struck, ensuring both accurate results and manageable computation.

1.3.3 Automatic parameter tuning

Table 1 - Compare the γ, σ^2, cost and runtime of two algorithms.

Algorithm	Gamma (*γ*)	Sig2(σ^2)	Cost	Time (s)
simplex	4.1972	0.4158	0.0400	0.4950
simplex	3.1185	0.3701	0.0400	0.4619
simplex	2.7710	5.1498	0.0400	0.4754
simplex	0.6605	4.8456	0.0400	0.5286
gridsearch	0.1113	4.4085	0.0300	0.9929
gridsearch	12.4450	0.1560	0.0400	0.9355
gridsearch	1.8769	0.0859	0.0400	0.9695
gridsearch	0.0779	1.1574	0.0300	0.8971

We evaluated two automated parameter tuning techniques: simplex and grid search. Simplex, a derivative-free method, approximates local optima, while grid search exhaustively trains parameter combinations to select the best error outcome. Comparing their performance in terms of measurement run time and cost, the results are presented in Table 1. Four runs were conducted for each algorithm, and we can observe that the gam and sig2 of the two methods and each run are quite different, which indicates that we have many local minima. Interestingly, despite different parameter values, the minimum cost remains similar across methods and runs. Notably, grid search proves approximately twice as slow as simplex due to its exhaustive nature and constrained search range. In conclusion, hyperparameter selection involves a non-convex problem with numerous local minima, yielding significant variability in optimized parameters from run to run.

1.3.4 Using ROC curves

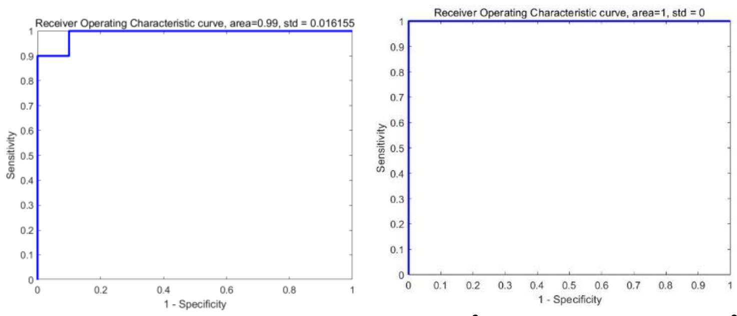


Figure 12 - The roc curves. (Left: γ =100, σ^2 =100, Right: γ =100, σ^2 =0.1)

Evaluating a system solely through accuracy isn't always suitable, especially when false positives and false negatives carry different costs. Stability also hinges on class balance in train-validation splits. To address this, ROC (Receiver Operating Characteristic) curves help portray a model's sensitivity and specificity. The ROC curve, plotted on the test set using tuned parameters, illustrates true positive rate against false positive rate. Maximizing the area under this curve (AUC) enhances classification quality.

In Figure 12, we compare ROC curves for identical gam and varying sig2 using the iris dataset. Notably, the right graph exhibits a larger AUC, signifying superior model performance. This analysis underscores the utility of ROC curves in assessing a model's ability to distinguish between classes.

1.3.5 Bayesian framework

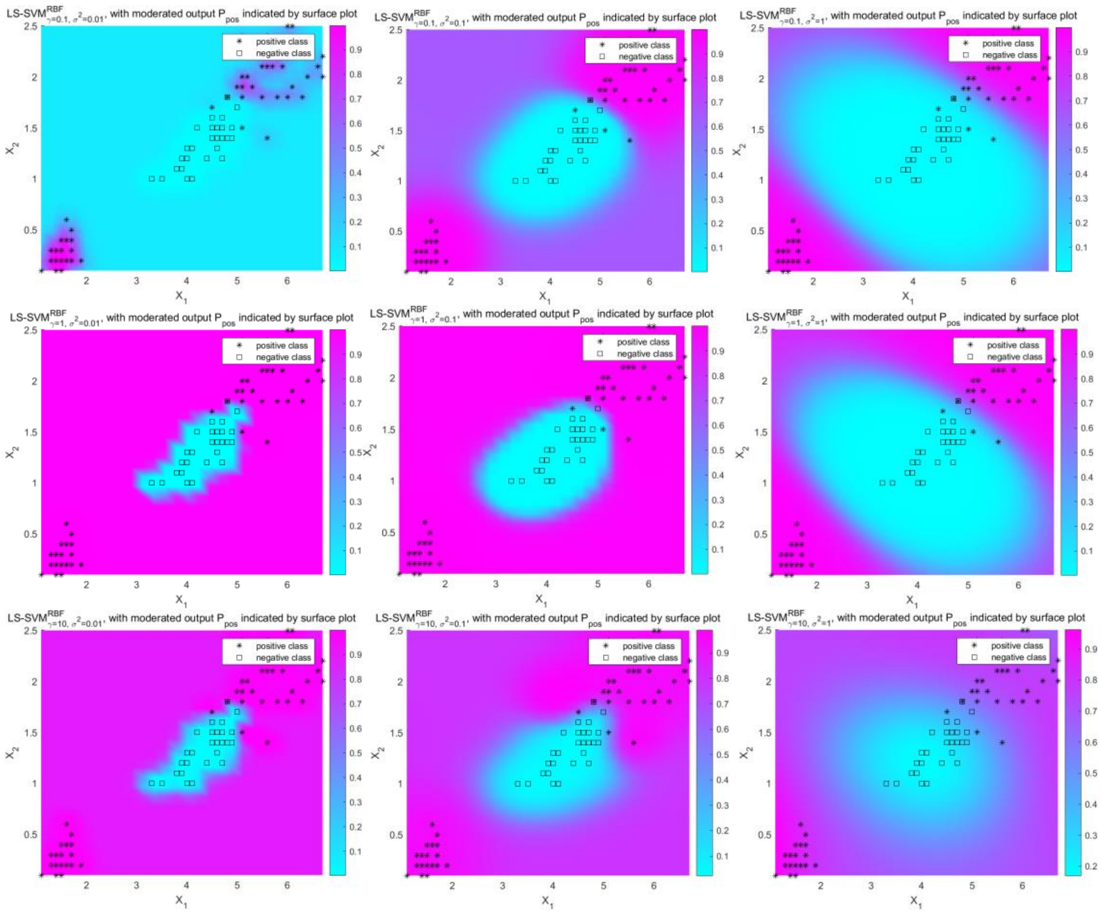


Figure 13 - Bayesian Inference with LS-SVM. (1st row, γ =0.1, 2nd row, γ =1, 3rd row, γ =10; 1st column, σ^2 =0.01, 2nd column, σ^2=0.1, 3rd column, σ^2=1)

Employing a Bayesian framework to maximize the posterior probability, we explore the effects of different gam and sig2 combinations. Figure 13 showcases gam and sig2 classification outcomes in individual subplots. The color scale reflects positive class probability: purple signifies higher, while blue signifies lower.

When sig2 = 0.1, performance excels, clear color differentiation between different classes and a smooth classification boundary are evident. Performance degrades when sig2 is excessively small or large, manifested by data points from distinct classes exhibiting similar colors. Similar observations apply to gam. This analysis reinforces the significance of parameter selection in the Bayesian context.

1.4 Homework problems

The Ripley dataset comprises two classes in two dimensions, featuring 250 training observations (125 per class) and 1000 test observations (500 per class). Figure 14 illustrates the dataset's distribution, revealing significant overlap in both dimensions. This suggests an optimal decision boundary should be multidimensional, rather than linear. The task involves LS-SVM classification across different kernels: linear, polynomial, and RBF. The decision boundaries of LS-SVM with different kernels are compared in Figure 15. Each ROC curve corresponding to the kernel is shown in Fig.16.

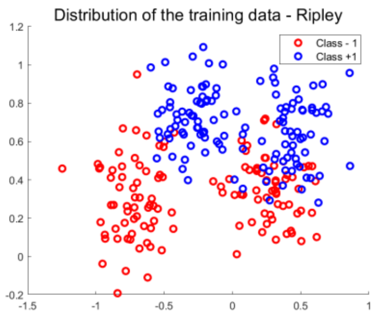


Figure 14 - The distribution of the training data of Ripley data.

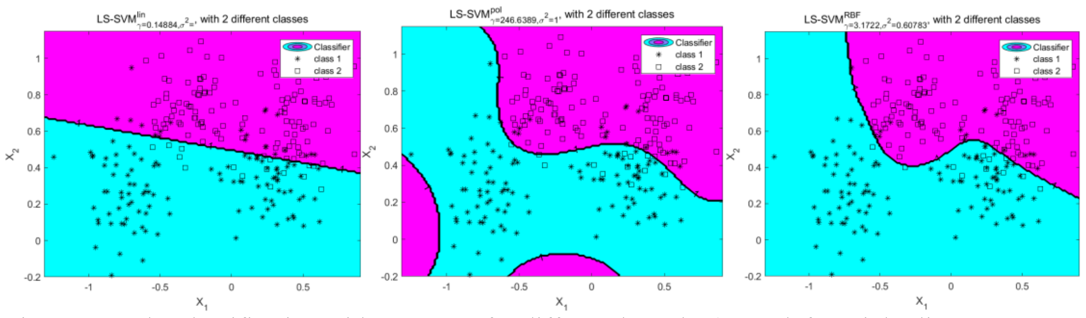


Figure 15 - The classification with LS-SVM for different kernels. (From left to right: linear, RBF, polynomial)

It can be seen from the classification diagram that LS-SVM with RBF kernel is the method to capture the best data distribution. In addition, from the ROC curve, the RBF kernel exhibits the highest AUC, though marginally outperforming the other kernels. In conclusion, the RBF kernel proves optimal for this dataset, effectively deconstructing its complex structure.

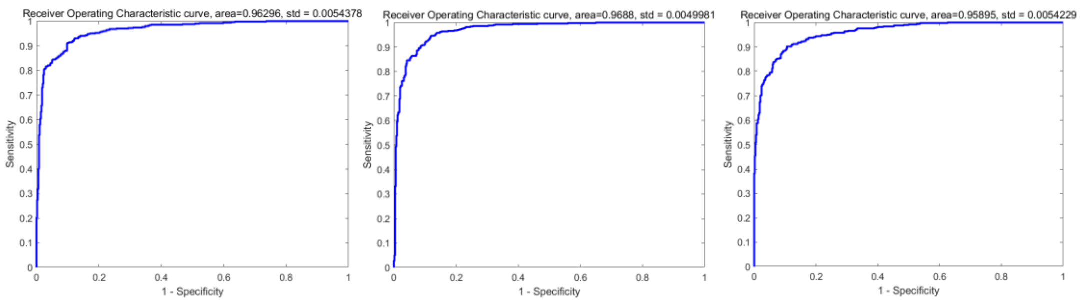


Figure 16 - ROC curves for different kernel with ripley dataset (From left to right: linear, RBF,

polynomial)

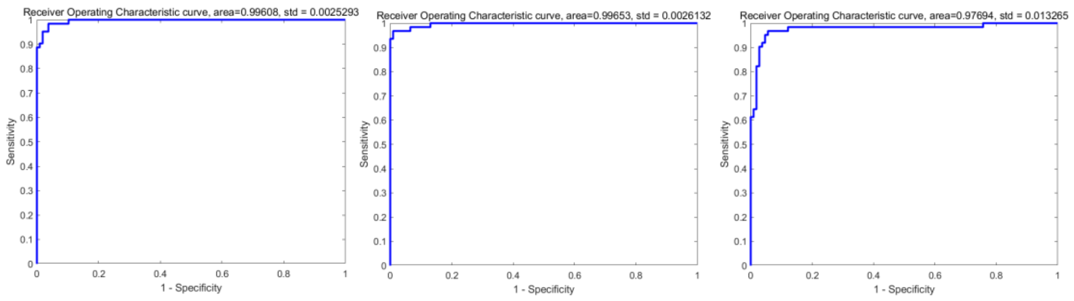


Figure 17 - ROC curves for different kernel with Wisconsin dataset (From left to right: linear, RBF, polynomial)

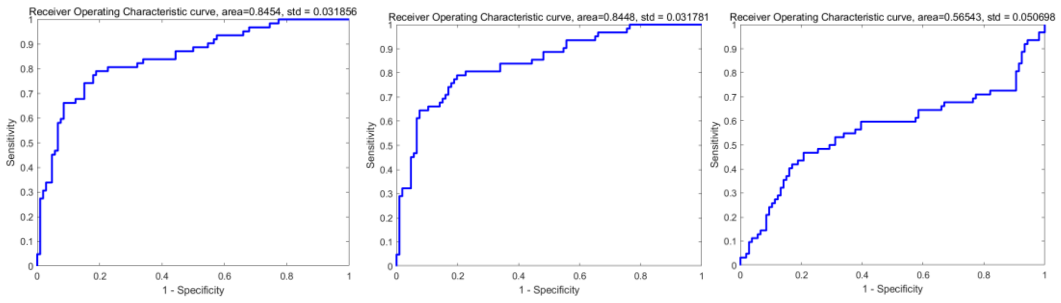


Figure 18 - ROC curves for different kernel with diabetes dataset (From left to right: linear, RBF, polynomial)

Considering the Wisconsin breast cancer dataset, the complexity lies in its higher dimensionality of 30. The dataset comprises 400 training points which contains 250 "-1" class, 150 "+1" class, and 169 test points which contains 107 "-1" class, 62 "+1" class. It is important to note that the dataset is imbalanced. Due to the intricacies of visualizing high-dimensional data, the focus here is solely on comparing ROC curves for different kernels. Figure 17 showcases these ROC curves. Impressively, all three kernels yield high AUC values, with linear and RBF kernels performing relatively better.

Transitioning to the diabetes dataset, featuring 8 dimensions, it consists of 300 training samples which contains 205 "-1" class, 95 "+1" class, and 168 test samples which contains 106 "-1" class, 62 "+1" class. This dataset is also imbalanced. Figure 18 reveals that the ROC curve performance is suboptimal overall. Notably, RBF and linear kernels exhibit higher AUC compared to the polynomial kernel.

To address the diabetes dataset's imbalance, techniques like introducing class weights in the objective function or leveraging stratified cross-validation can be employed to rebalance the data. These strategies adjust class representation to enhance classification performance.

2. Assignment 2: Function Estimation and Time Series Prediction

2.1 Support vector machine for function estimation

- *Linear data*

In SVM function estimation, the parameter "e" signifies the insensitive region where points are tolerated and not penalized. A larger "e" allows for greater allowable errors. In the linear case, the formula for linear SVM regression is ŷ = w^T x + b, where "w" and "b" are determined under the constraints within the "e" tube. For non-linear cases, the support vectors exist outside this tube, hence "e" strongly influences the regression line.

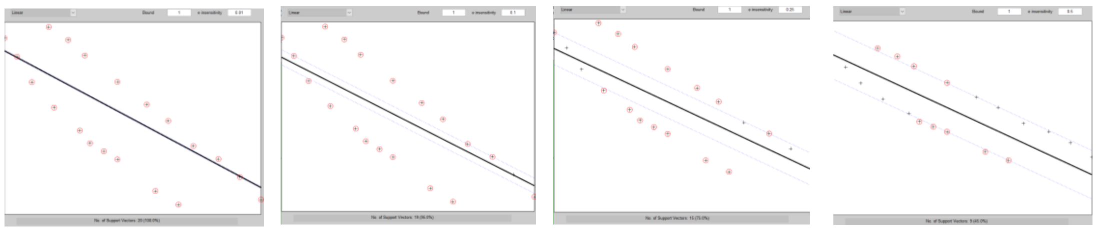


Figure 19 - The SVM regression with fixed bound (bound =1) and different e (from left to right: 0.01, 0.1, 0.25, 0.1)

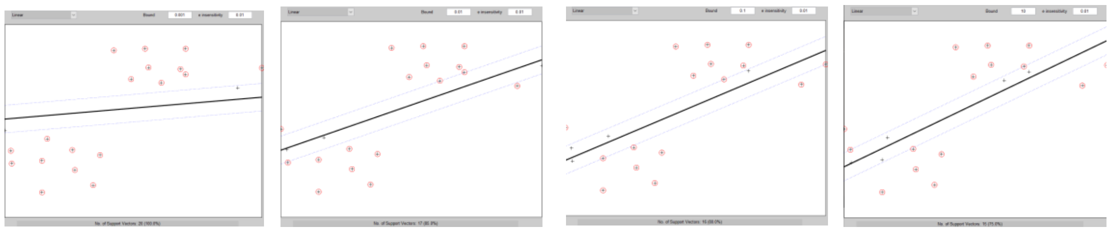


Figure 20 - The SVM regression with fixed e (e =0.01) and different bound (from left to right: 0.001, 0.01, 0.1, 10)

Figure 19 illustrates SVM regression with fixed bounds (bound = 1) and varying "e". As "e" increases, the margin widens, and fewer support vectors are found outside the edge. Figure 20 presents SVM regression with a constant "e" (e = 0.01) and varying bounds. It is apparent that as the bound increases it restricts weight values.

In essence, "e" determines error tolerance in SVM regression, while bounds influence weight constraints. These observations emphasize the dynamic interplay of parameters in shaping the behavior of SVM regression.

- *More challenging dataset*

When faced with nonlinear data, the inadequacy of the linear kernel prompts exploration of other kernels provided by the system. Through manual parameter selection and examination, the goal is to identify the most suitable regression function for the data.

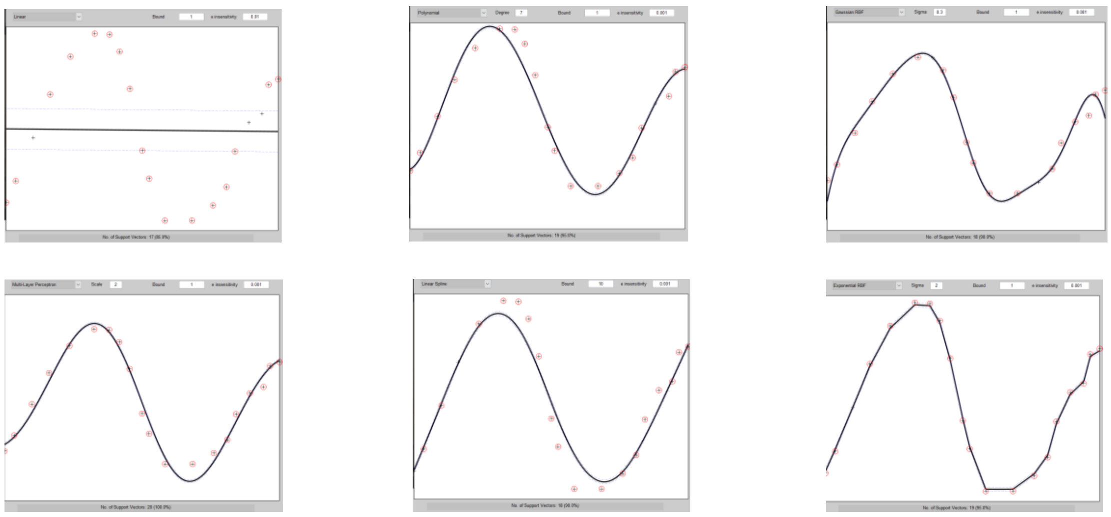


Figure 21 - The SVM regression with different choices of kernel. (From left to right, top to bottom: linear, polynomial, gaussian RBF, multi-layer perceptron, linear spline, exponential RBF)

Table 2 - Compare the performance of different kernels.

Kernel	Linear	Polynomial	Gaussian RBF	Multi-layer perceptron	Linear spline	Exponential RBF
NO. of Support vectors	17	19	18	20	18	19
Accuracy	85%	95%	90%	100%	90%	95%

The outcomes, presented in Figure 21, encompass six kernels: linear, polynomial, Gaussian RBF, multi-layer perceptron, linear spline, and exponential RBF. Refer to Table 2 for comprehensive details. The linear kernel registers the lowest performance, while the others perform well, boasting accuracy rates surpassing 90%. The polynomial kernel and linear spline both exhibit a minor deviation for the peaks, while the Gaussian RBF and multi-layer perceptron kernels excel at capturing nonlinear patterns.

A fundamental distinction between SVM regression and classical least squares fitting lies in their distinct loss functions and SVM regression uses only a subset of the data to define the regression line. Least squares fitting employs the L2 norm, seeking the straight line with the smallest distance from observations, conversely, SVM linear kernel uses the L1 norm, which points in the insensitive region produce no error.

2.2 A simple example: the sinc function

2.2.1 Regression of the sinc function

A LS-SVM regression model is established using the RBF kernel, with parameter tuning done both manually and through an algorithm. The process involves experimenting with different γ and σ values and evaluating Mean Squared Error (MSE) to identify optimal solutions.

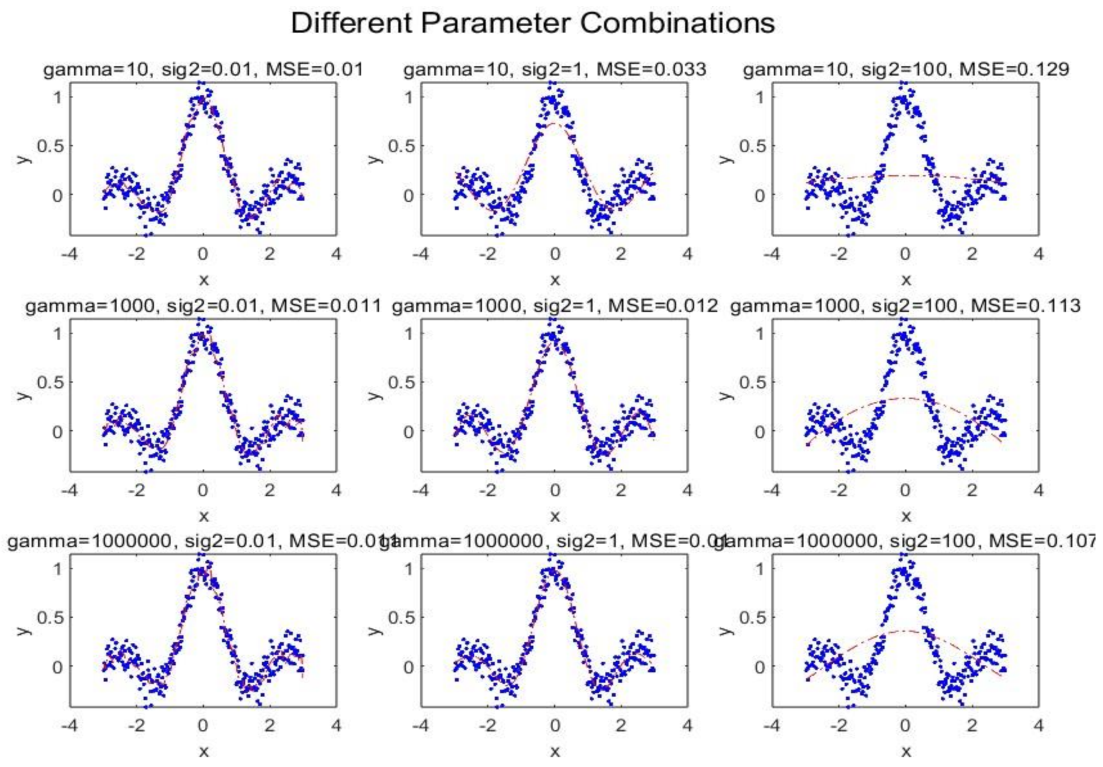


Figure 22 - a range of different, γ and σ^2 of LS-SVM regression with RBF kernel with MSE

Table 3 - Compare the MSE of different combinations of γ and σ^2

Gamma (γ)	**Sig2 (σ^2)**	**MSE**
10	0.01	0.0105
10	1	0.0327
10	100	0.1286
10^3	0.01	0.0108
10^3	1	0.0119
10^3	100	0.1130
10^6	0.01	0.0111
10^6	1	0.0099
10^6	100	0.1073

Figure 22 portrays outcomes with varying gam (10, 10^3, 10^6) and sig2 (0.01, 1, 100). Notably, when sig2 is 0.01 and 1, the fitting quality is comparatively superior. This observation is reinforced by Table 3, where other gam-sig2 combinations exhibit similar fits. Remarkably, the pair with σ=0.10 and γ=10 yields the lowest MSE of 0.0105.

However, the pursuit of the optimal hyperparameter pair encounters challenges due to the non-convex nature of selecting the best combination based on MSE. Local minima proliferate, without a singular global minimum. An optimal model should account for both fit and complexity, striking a balance between the two.

To automate LS-SVM tuning, the "tunelssvm" method was employed, utilizing miss-one cross-validation. Two distinct techniques were compared: simplex and grid search. The simplex algorithm approximates local optima without derivatives, while grid search involves training models with varying parameters and selecting the configuration yielding the smallest error.

Table 4 - Automatic tuning of the parameters

Algorithm	**Gamma (γ)**	**Sig2 (σ^2)**	**MSE**
simplex	70.0223	0.311099	0.0105
	205.4931	0.435169	0.0103
	23.2896	0.362592	0.0103
gridsearch	42.6972	0.273123	0.0105
	2022.8701	0.53761882	0.0104
	5590.4607	0.76998194	0.0102

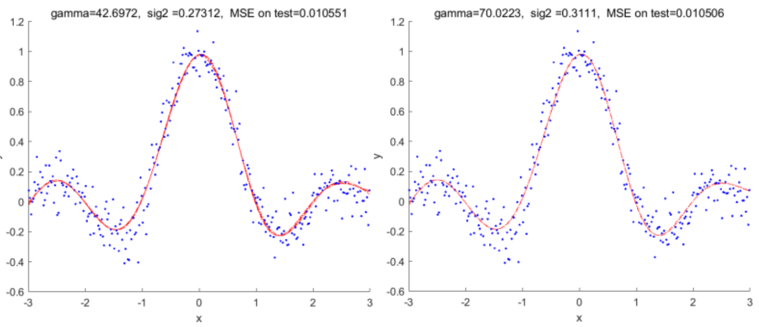


Figure 23 - Tuning the parameters with LS-SVM regression using Simplex (Left) and Grid search (Right)

Figure 23 showcases the results, revealing highly effective and comparable fitting outcomes for both techniques. Corresponding data in Table 4 indicates similar MSE values around 0.010. Across multiple runs, the results remain consistent, with the grid search method consuming more time due to the non-convex nature of parameter tuning. In essence, the intricacies of parameter tuning become evident once again: the non-convex problem landscape yields local minima as potential choices.

2.2.2 Application of the Bayesian framework

A Bayesian framework offers a valuable avenue to fine-tune and analyze LS-SVM regressors, structured through a three-tiered process wherein each tier's evidence informs the next. The journey commences with initializing parameters a and b in the first layer. Progressing to the second layer, the focus shifts to optimizing the regularization parameter gam, followed by the third layer's optimization of the kernel parameter sig2. The overarching objective is to identify parameters that maximize the posterior probability. In Figure 24, the outcome of this tuned regression function is showcased. Beginning with initial parameter values (sig2 = 0.4, gam = 10), a sequence of iterations culminates in the attainment of optimized parameters (gam = 0.52892, sig2 = 0.21135).

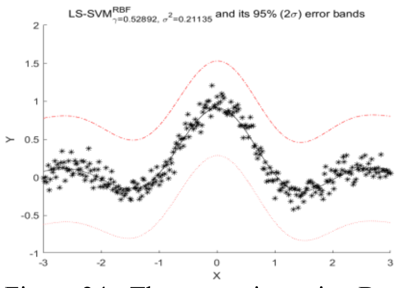


Figure 24 - The regression using Bayesian Framework with 95% confidence.

2.3 Automatic Relevance Determination

Automatic relevance determination is a technique employed to discern the most influential subset of inputs, minimizing costs. Distinct dimensions are assigned varying weight parameters. Through reverse selection, the dimension with the largest optimal sig2 is progressively eliminated. The dataset, encompassing 100 rows and 3 dimensions, is utilized for this purpose. Parameters previously tuned are employed, and the process identifies the first dimension as the most crucial.

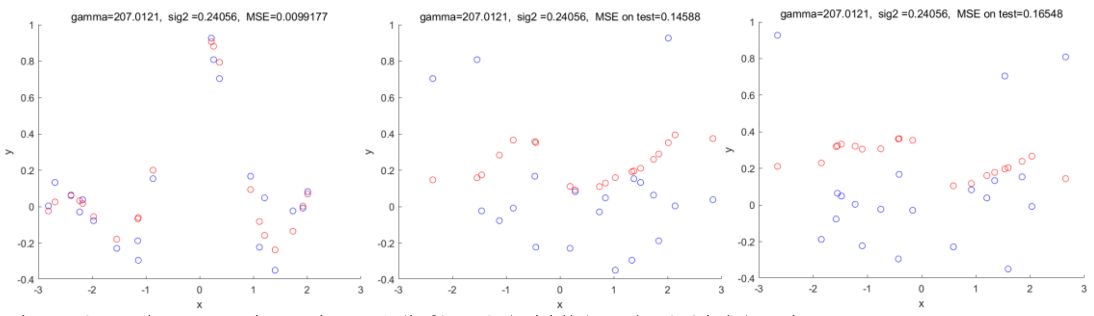


Figure 25 - The regression using X1 (left), X2 (middle) and X3 (right) as input.

able 5 - The relevance rank and MSE of the inputs

Input	Relevance rank	MSE
X1	1	0.0099
X2	2	0.1458
X3	3	0.1655

In confirmation, an LS-SVM is trained using only one input per dataset dimension. As revealed in Table 5, relevance ranks are determined for the three inputs. X1 emerges as the most relevant, substantiated by the lowest MSE. X2 follows, while X3 lags in relevance. Regression results depicted in Figure 25 validate this inference. The optimal parameters obtained are gam = 207.0121, sig2 = 0.2405571.

Comparing models with different input combinations, cross-validation can measure generalization performance through validation set errors. Alternatively, feature selection can involve iteratively excluding one feature at a time, calculating cross-validated MSE for the rest of the dataset. The highest MSE corresponds to the most relevant feature. This approach offers a robust strategy for feature selection.

2.4 Robust regression

An acknowledged drawback of LS-SVM is its vulnerability to outliers due to the squared loss function. To address this, robust solutions such as Huber loss function or weighted LS-SVM are introduced. Weighted LS-SVM assigns weights (vk) to each data point, enabling control over the impact of outliers on the loss function. When outliers exist, it assigns these outliers a lower vk to limit their impact on the loss function.

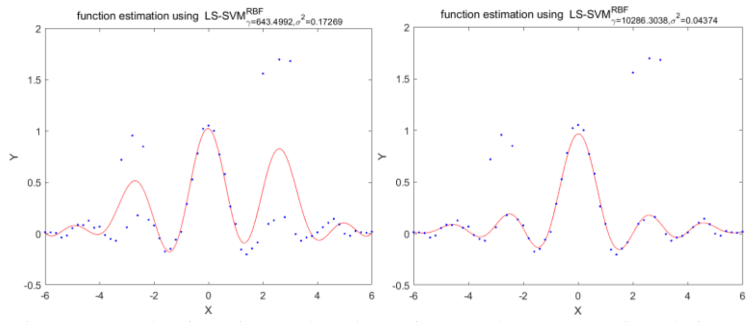


Figure 26 - The function estimation of non-robust regression (left) and robust regression (right).

Figure 26 compares the unrobust LS-SVM version with the robust version (utilizing Huber's method) to visualize the difference in regression outcomes. Notably, outliers above the left and right peaks considerably impact non-robust regression, often leading to overestimated function estimates. Conversely, robust LS-SVM remains largely unaffected by these outliers, delivering enhanced performance.

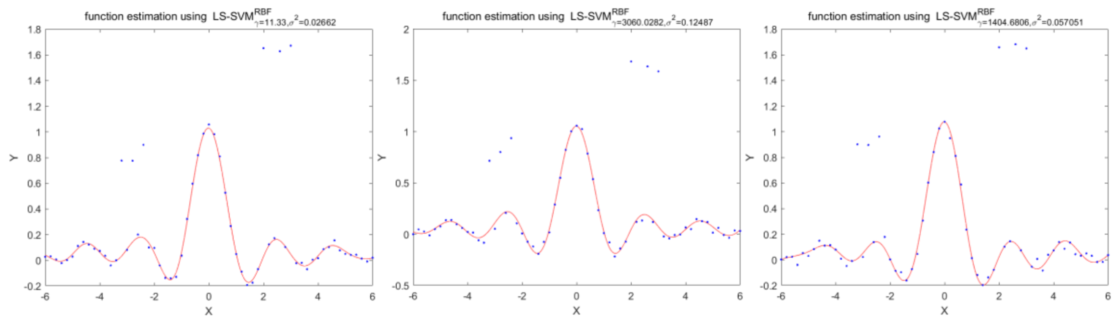


Figure 27 - Regression with weighted robust LS-SVM with weighting function Hampel function (left), Logistic function (middle), and Myriad function (right).

Subsequently, the influence of assigned weights on regression values is evaluated across four methods: 'whuber', 'whampel', 'wlogistic', and 'wmyriad'. Regression plots for 'whuber' are displayed in Figure 26, while Figure 27 showcases plots for the remaining three methods. All four methods exhibit robust to outliers, with their estimated functions show similarity. To assess fitting quality, Mean Absolute Error (MAE) is computed, with the following values: Huber = 0.13267, Hampel = 0.12729, Logistic = 0.13302, and Myriad = 0.14054. Consequently, the regression results of the Hampel method yield the best outcome. This investigation underscores the potency of robust approaches, such as weighted LS-SVM with Hampel's method, in mitigating the influence of outliers on regression outcomes.

Here Mean Absolute Error (MAE) is preferred over classical Mean Squared Error (MSE) because MSE is more affected by outliers. Because MSE will give more weight to distances, and outliers are always far away from other points, MSE will give more weight to outliers than MAE when running a regression.

2.5 Homework problems

2.5.1 Introduction: time series prediction

Time series forecasting involves predicting the next data point in a sequence based on preceding data. Achieving optimal performance for LS-SVM in this context requires tuning parameters such as gam, sig2, and order. The simplex method tunes sig2 and gam, employing 10-fold cross-validation for assessment using Mean Absolute Error (MAE).

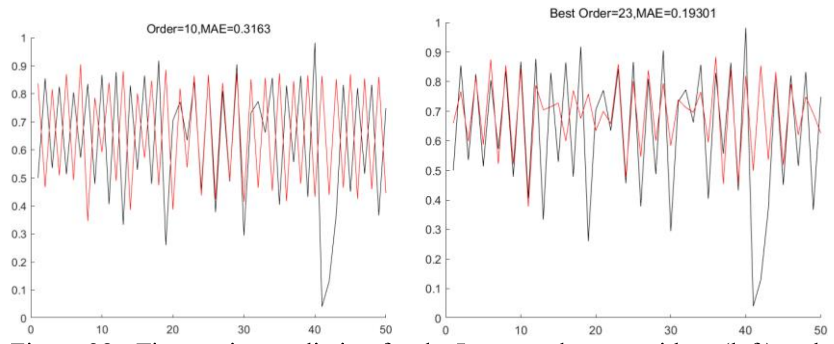


Figure 28 - Time series prediction for the Logmap dataset, without(left) and with tuning (right)

For the Logmap dataset, encompassing 150 data points with a goal of predicting the next 50, the ideal order parameter—discovered through optimization—is approximately 23. This order parameter is subsequently used for optimizing gam, sig2, and order. The optimized MAE of 0.19301 significantly improves upon the unoptimized MAE of 0.3163.

Comparing unoptimized and optimized time series predictions with order values of 10 and 23 respectively, Figure 28 highlights a substantial difference. The curve fitting with an order value of 10 yields inadequate results, failing to capture sharp changes across different time points. Conversely, an order value of 23 generates a relatively flat fitting curve with limited predictive capacity.

2.5.2 Santa Fe dataset

The Santa Fe dataset, encompassing 1000 time points with a prediction goal of the next 200 points, necessitates parameter tuning. Given significant value fluctuations between adjacent time points, Mean Absolute Error (MAE) rather than Mean Squared Error (MSE) is employed for assessment.

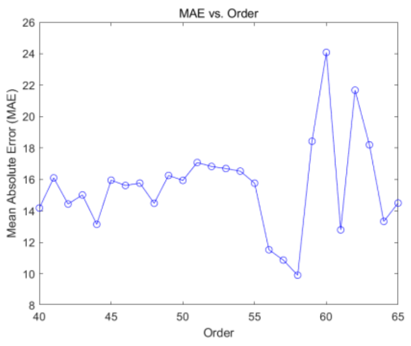


Figure 29 - The visualization of MAE of different orders.

Visualizing the relationship between different orders and their corresponding minimum MAE values in Figure 29, it's evident that the global minimum MAE aligns with order=58. Consequently, order values of 50 and 58 are selected for prediction. To determine which order offers superior predictive ability, time series prediction plots are generated in Figure 30.

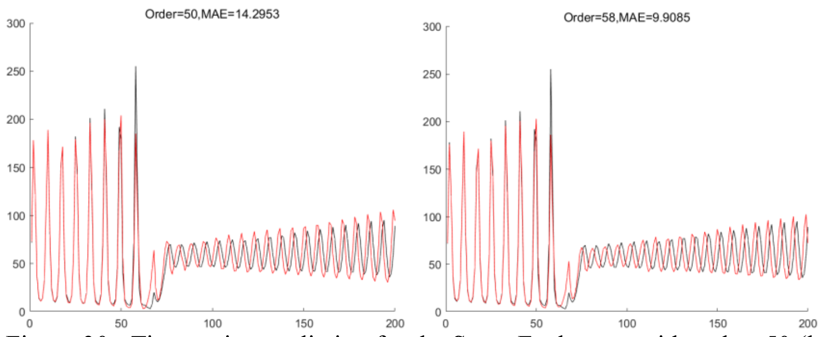


Figure 30 - Time series prediction for the Santa Fe dataset, with order=50 (left), and with the optimal order=58

Observing Figure 30, predictions for both order 50 and 58 closely match real data up to time point 60. After this point, predictions for order 50 begin deviating more significantly than those for order 58. This contrast becomes more pronounced when comparing their MAE values. The MAE of 14.2953 for the prediction with order=50 significantly improves to 9.9055 for the prediction with order=58. This compellingly demonstrates the efficacy of tuning in producing more accurate forecasts.

However, the predicted result of order=50 has been greatly improved compared to the untuned prediction result (e.g.: order=10). Furthermore, leveraging a verification set for parameter optimization is viable. The data splitting process—into training, verification, and test sets—must ensure continuity of time points to avert overfitting.

3. Exercise Session 3: Unsupervised Learning and Large Scale Problems

3.1 Kernel principal component analysis

Principal Component Analysis (PCA) hinges on identifying the direction of maximal variance by extracting the leading eigenvector from the covariance matrix of data distribution. Larger eigenvalues signify pivotal data components, while negligible ones typically denote noise. Consequently, dimensionality reduction and denoising can be achieved by retaining prominent eigenvalues.

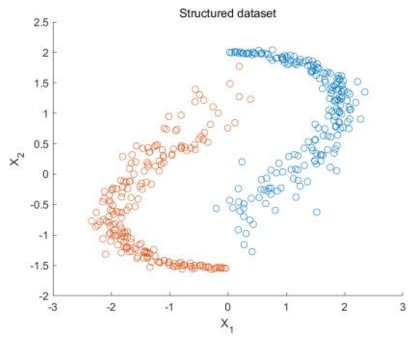


Figure 31 - The distribution of original dataset. Color represents different classes.

Figure 31 visualizes the original dataset distribution, with distinct colors denoting different classes. Evidently, this nonlinear dataset comprises two classes. Next, kernel PCA is employed to denoise the data using varying numbers of principal components (PCs). Selecting an optimal number of PCs is vital: excessively few components may result in excessive information loss, whereas an overabundance may inadequately eliminate noise. For this analysis, 2, 6, and 10 principal components are considered.

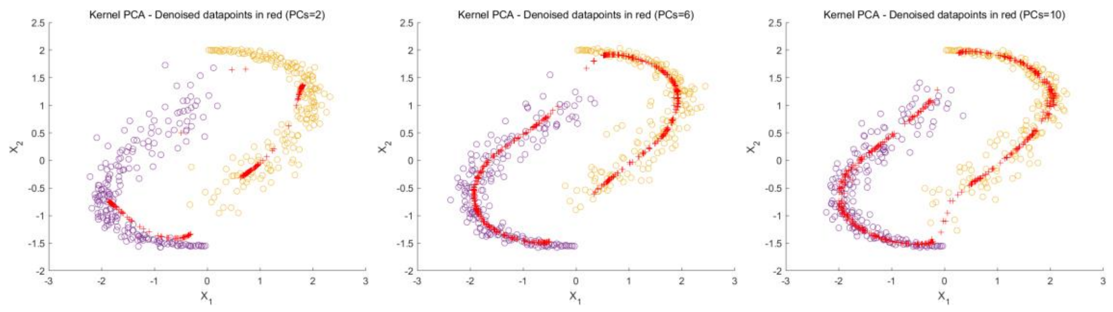


Figure 32 - Denoising the data using the kernel PCA with different number of PCs (from left to right: PCs=2, 6, 10)

Figure 32 shows denoising the data using the kernel PCA with different numbers of PCs. When employing only 2 components, the denoised data fails to capture the data structure comprehensively. Conversely, using 10 components still permits substantial noise influence. Opting for 6 components strikes a balance, effectively representing the data structure while mitigating noise impact.

To summarize, kernel PCA presents a powerful technique for dimensionality reduction and denoising, with the choice of the right number of principal components playing a crucial role in achieving a harmonious trade-off between data structure representation and noise removal.

- *Linear PCA VS Kernel PCA*

In dealing with nonlinear data distributions, linear PCA encounters limitations. Consider the Yin-Yang dataset depicted in Figure 31. Linear PCA, designed for linear relationships, struggles to capture the true direction of maximum variance, as evident in Figure 33.

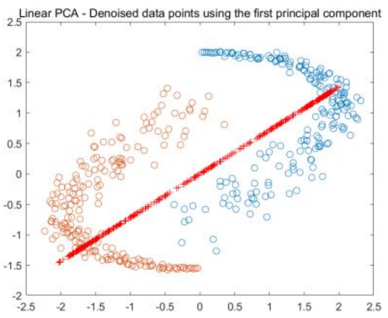


Figure 33 - Denoised data with linear PCA.

Linear PCA's maximum number of principal components equals the original dimension count. In this case, the 2-component limit results in subpar analysis, limiting its effectiveness. Kernel Principal Component Analysis (PCA) offers a solution by leveraging kernel functions to project data into higher-dimensional feature spaces. The maximum number of principal components in kernel PCA is determined by the dimension of the kernel matrix, which, in this instance, is 400.

- *Tuning of the parameters*

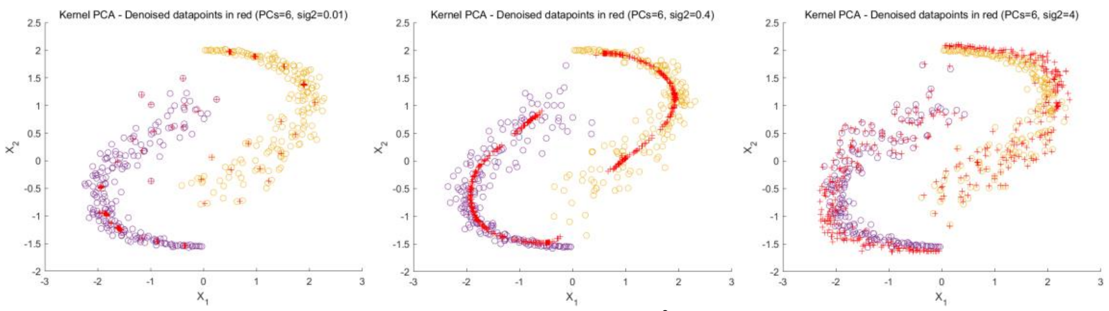


Figure 34 - The kernel PCA with different kernel parameter σ^2 (from left to right: 0.01, 0.4, 4)

For the dataset we first tuned the parameters for sig2. Various sig2 values are tested, and the kernel PCA result yielding the smallest error between denoised and expected modes is selected. Figure 34 illustrates the outcome of this process. When maintaining 6 principal components and varying sig2, an optimal choice is evident. Sig2=0.4 yields well-denoised data points with clear data structure representation. However, as sig2 increases to 4, the model accommodates more noise, resulting in overfitting. Conversely, sig2 of 0.01 filters out too many details, obscuring the complete data structure.

3.2 Fixed-size LS-SVM

The choice between primal and dual formulations depends on the following points. The original formulation is preferred when 1) the dataset has many samples but relatively few features, as this can be computationally faster than the dual problem, 2) when a linear kernel is used and the data is well balanced between classes, and 3) if the expected solution is sparse. The dual formulation is preferred when 1) you have a small number of samples in high dimensions, and 2) when dealing with non-linearly separable data using kernel tricks (e.g., using polynomial or RBF kernels), the dual problem involves dealing with the inner product of data points, which in the data Especially useful when mapping to high-dimensional spaces.

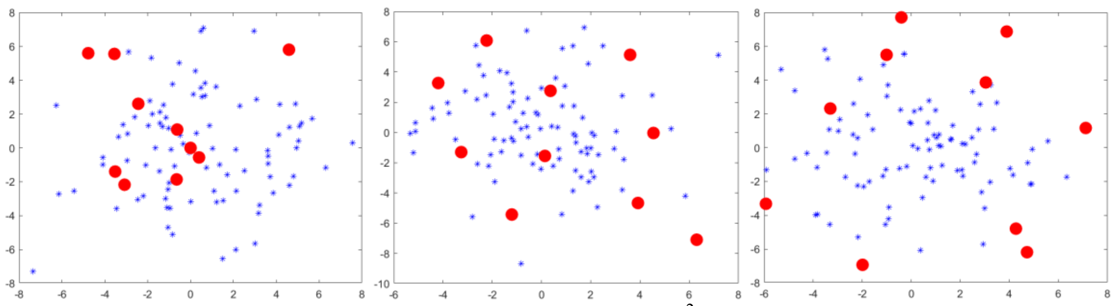


Figure 35 - Selected subset M of support vector with different σ^2 (from left to right: 0.001, 1, 1000)

The chosen kernel parameter, sig2, decides the process of selecting a representative subset. This subset, comprised of support vectors, contains the distinctive traits of various regions in the dataset, laying the foundation for constructing a robust model within the feature space.

In Figure 35, the influence of sig2 on the selected subset M of support vectors becomes evident. As the value of sig2 escalates, the chosen support vectors disperse more widely across space. Notably, when sig2 is set to a low value like 0.001, discerning the inherent data distribution characteristics proves challenging. Conversely, when sig2 is elevated to a high value such as 1000, the number of support vectors balloons, nearly approaching the dataset's size. Regrettably, this dilutes the potential benefits derived from using a more compact subset of data. A sweet spot emerges around sig2 = 1. This value strikes a balance, effectively representing the intricate data distribution while maintaining a manageable number of support vectors. This balance contains the essence of the data, facilitating the construction of a well-fitted model.

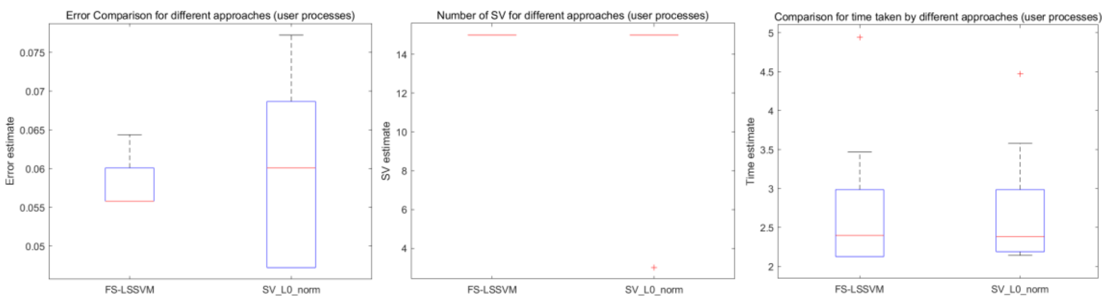


Figure 36 - The error, the number of support vector and time taken for the Fixed size LSSVM and l0-approximation on the Shuttle dataset.

Figure 36 offers a comparison between the fixed-size LS-SVM and the l0-approximation method using the Suttle dataset. We assess their performance in terms of error rate, computational time, and the count of support vectors employed. Notably, the number of support vectors and the computational time exhibited by both methods show a remarkable similarity. Furthermore, we observe that the error rate of the fixed-size LS-SVM is marginally lower compared to the l0-approximation technique. It’s crucial to acknowledge that the outcomes can vary across different runs, which introduces an element of variability that might impact our analysis.

3.4 Homework problems

3.4.1 Kernel principal component analysis

In Figure 37, a comparison is made between the denoising capabilities of linear PCA and KPCA using digitized handwriting images. The chosen noise factor is 1.0, while the KPCA parameter is set to sig2=35.9078. Upon closer inspection, it's evident that the presence of substantial noise in the second row of numbers renders them indecipherable.

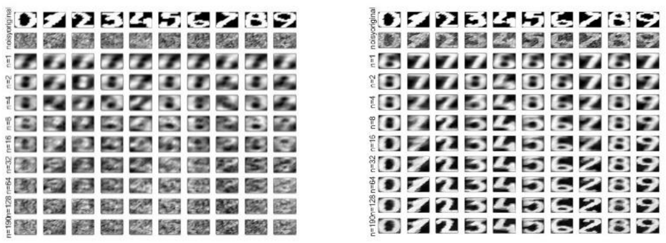


Figure 37 - The digital handwriting image denoising with linear PCA and PCA kernel with RBF kernel.

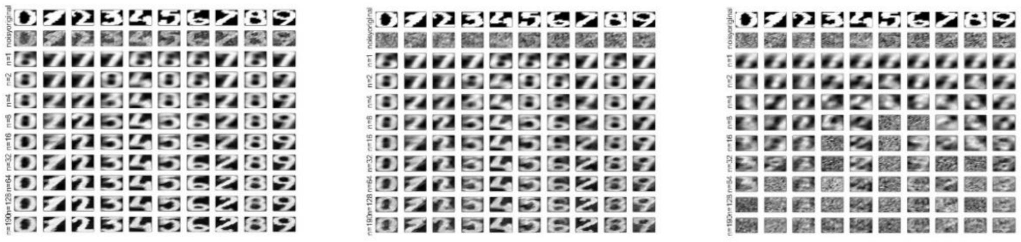


Figure 38 - Denoising using KPCA, for different Sigma Factor values (from left to right: 0.01, 1, 100)

When employing linear PCA with 2, 4, or 8 extracted components, the denoising outcomes appear optimal among the different component counts. Despite this, the resulting numbers remain unclear for accurate recognition. Often, KPCA-derived components lead to more successful denoising results, notably achieving accurate digit pattern identification after the extraction of just 4 components. This is attributed to the ability of Kernel PCA to capture non-linear patterns of maximum variance.

Figure 38 demonstrates the impact of the Sigma factor parameter. With sigma factor values of 0.01, 1, and 100 tested, a trend emerges wherein smaller sigma values result in stronger denoising capability. However, excessively small values may lead to complete noise removal but introduce extrapolation artifacts, compromising accuracy. Conversely, higher sigma values struggle to effectively eliminate sufficient noise. A favorable trade-off is observed with a sigma factor of 1 and n values of 16 or 32, offering promising denoising outcomes.

- *Tuning of parameters*

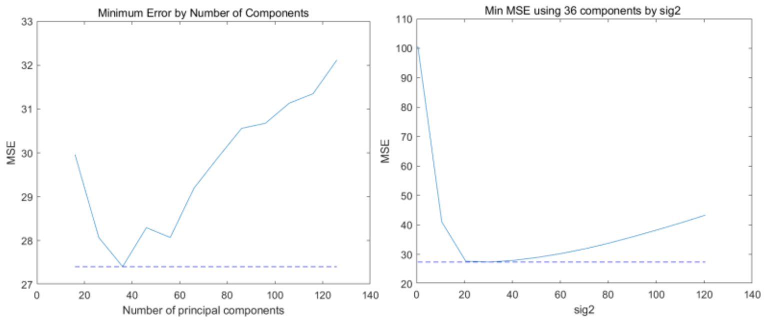


Figure 39 - The tuned minimum error by CPs (left) and minimum MSE by σ^2 (right) for Xtest1.

The parameter tuning involves adjusting sigma and the number of principal components considered. Beginning with a noise factor of 1.0 and a sigma factor of 0.5 based on prior experience, the process is initiated. First, the minimum error is determined by varying the number of components (CPs) and plotting their corresponding mean squared error (MSE), as depicted in Figure 39, and the CP value corresponding to the lowest MSE is selected. Subsequently, evaluate different sig2 values and their corresponding MSEs (Figure 39), and select the sig2 value with the minimum MSE as the tuned sig2. For Xtest1, the optimal parameter after tuning is CPs=36, sig2 =30.5. The same steps are applied to Xtest2, and the best parameters after tuning can be obtained CPs=126, sig2 =10.5.

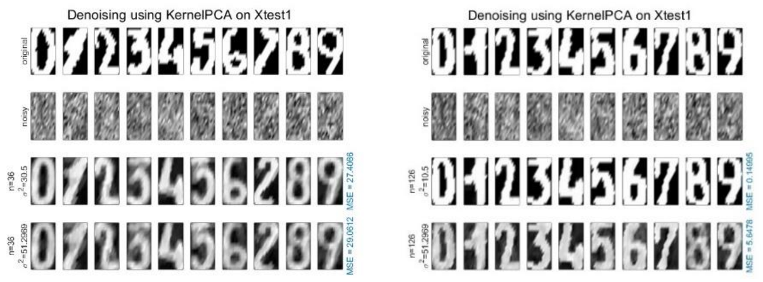


Figure 40 - The denoised images of Xtest1 (left) and Xtest2 (right) with tuned sigma factor and CPs.

Moving forward, the optimized CPs and sig2 are employed to compare the denoised images of Xtest1 and Xtest2 (Figure 40). The third row showcases images denoised with the fine-tuned parameters, while the fourth row presents denoised images using the default settings for CPs and sig2. The improved denoising efficacy resulting from tuning is evident. Notably, for Xtest2, the image reconstructed using the optimized parameters in the third row outperforms the default configuration in the fourth row, achieving enhanced accuracy and clarity.

3.4.2 Fixed-size LS-SVM

- *Shuttle dataset*

The Shuttle dataset comprises 58,000 observations categorized into 7 classes, with 9 numerical attributes. Notably, approximately 80% of the samples are from the first class, while the remaining 20% belong to the other 6 classes.

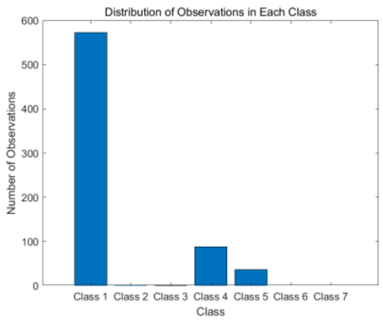


Figure 41 - The distribution of shuttle dataset in each class

The class distribution is depicted in the histogram (Fig. 41), highlighting the substantial prevalence of class 1 observations, with relatively fewer data points in classes 2, 3, 6, and 7. This skewed class distribution presents a distinct classification challenge, particularly for methods sensitive to class imbalance.

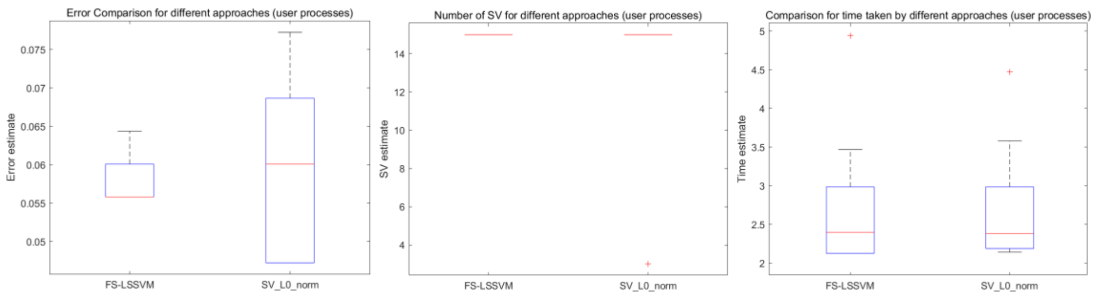


Figure 42 - The error, the number of support vector, and time taken for the Fixed size LSSVM and l0-approximation on the Shuttle dataset.

Due to the dataset being too large, this analysis considers only the first 700 samples. Linear check data is employed for classification, aiming to segregate data belonging to the first class from those in other classes. This results in 572 samples belonging to class 1 and 128 samples in other classes. A comparison between the fixed-size LS-SVM and L0_norm approximation methods is presented in Figure 42. In terms of runtime and the number of support vectors, both methods yield similar outcomes. A notable distinction arises in the error estimation. The L0_norm approximation exhibits a higher error value accompanied by greater variance. This indicates that the L0_norm approximation method introduces more variability in error estimates, which could impact its reliability.

- *California dataset*

The California dataset consists of 20,640 samples, featuring 9 attributes encompassing 8 independent variables and one dependent variable (ln(median home value)). The attributes encompass Median Income, Median Age in Housing, Total Rooms, Total Bedrooms, Population, Households, Latitude, and Longitude. The distribution of ln(median house price) is illustrated in Figure 43.

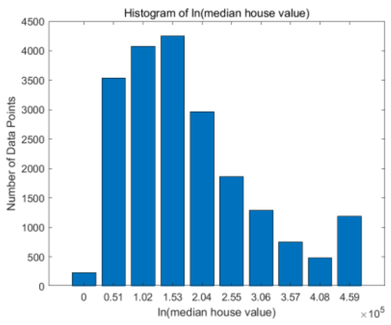


Figure 43 - The distribution of California dataset in ln(median house value)

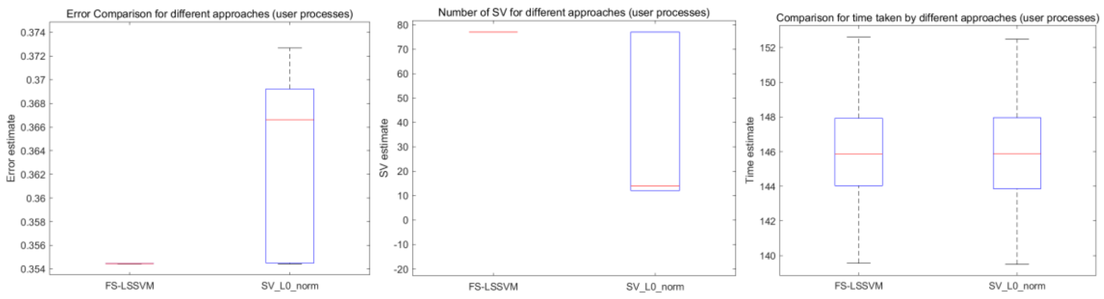


Figure 44 - The error, the number of support vector, and time taken for the Fixed size LSSVM and l0-approximation on the California dataset.

In this analysis, the RBF kernel is employed for both the fixed-size LS-SVM and the L0_norm approximation method. The comparison outcomes of these two methods on the California dataset are depicted in Figure 44. It's evident that the time taken by both methods is relatively similar. However, the error rate of the fixed-size LS-SVM is significantly lower in contrast to the L0_norm approximation method. Moreover, a distinct difference is observed in the number of support vectors used by the two methods, with the fixed-size LS-SVM having a higher count.

This discrepancy in results can be attributed to the fundamental nature of the L0_norm approximation method, which aims to minimize the number of relevant features or variables. This reduction in variables might lead to underfitting, causing a higher error rate and less accurate predictions. Conversely, the fixed-size LS-SVM employs a broader set of support vectors, which might enhance its predictive capacity and result in lower error rates. The selection between these methods should be driven by a thorough consideration of the specific dataset characteristics and modeling objectives.

Use of ChatGPT (or any other AI Writing Assistance) – Form to be completed

Student name: Zhiqi Wang

Student number: r0822618

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

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

Course name: Support Vector Machine ....

Course number: [H00H3a] ....

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.

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

....

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

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

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

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

- *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**

- *Code of conduct*: This use is comparable to e.g. a google scholar search. However, be aware that 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

X **To let generate programming code**

- *Code of conduct*: Correctly mention the use of ChatGPT 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 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, 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 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 template samen met de code of conduct zal in de toekomst nog verdere aanpassingen behoeven. Het schept alvast een kader voor de 2de en de 3de examenperiode van 2022-2023.