Support Vector Machines: Methods and Applications

Report

Vincent Buekers
r0754046

Master of Statistics
H02D3A
2019-2020

Session 1: Classification

1.1 Exercises

1.1.1 Two Gaussians

Consider a binary classification problem in 2 dimensional space (Fig. 1.1) with class distributions that have matching covariance structures. As depicted in the figure, a separating line can already lead to an accurate classification, may it not yet be perfect due to the overlap between the class distributions. Given the similarity of the covariance matrices, a linear decision boundary is in fact the optimal solution, much like one would obtain with a Fisher Discriminant analysis.

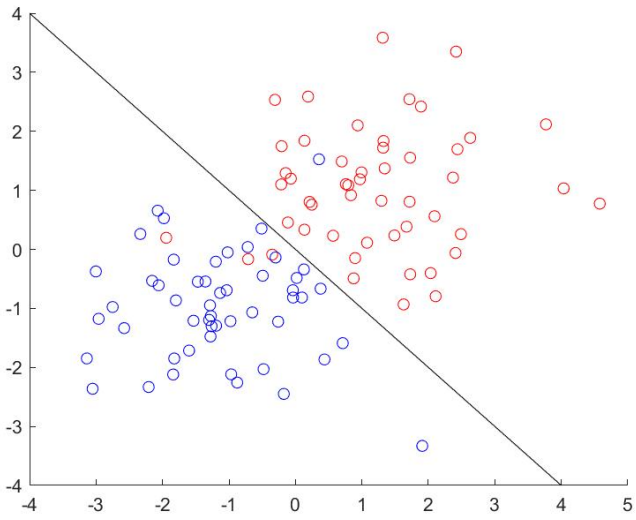
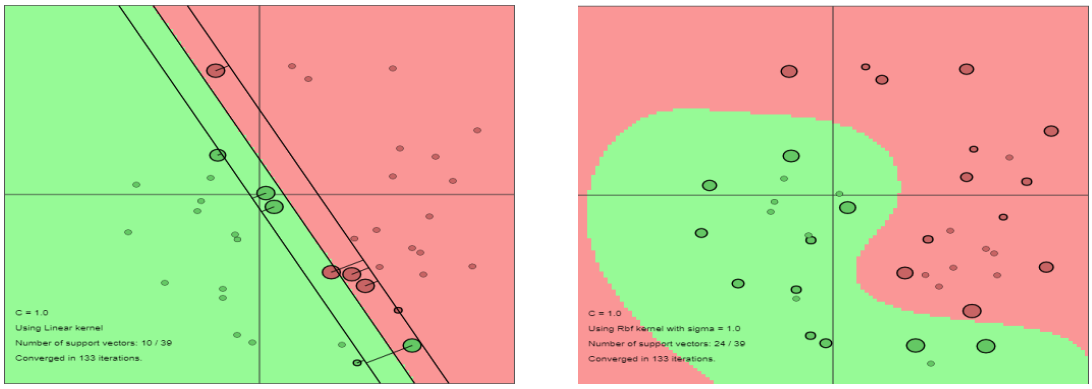


Figure 1.1: Two Gaussians

1.1.2 SVM Classifier

Support vectors are the unique sparse solution to a quadratic programming (QP) problem. That is, they are training data points that correspond to non-zero Lagrange multipliers α_k. However, not all support vectors are of equal importance since the data points vary in their contribution to obtain the solution. More specifically, the importance of a support vector is proportional to the size of the corresponding α_k. Geometrically, this translates to how close they are located to the decision boundary. Indeed, it can be seen that the points located closest to the decision boundary are most important as indicated by their circumference (Fig. 1.2, a). For a non-linear model, as obtained with an RBF kernel for example, the support vectors are also most closely located to the non-linear decision boundary (Fig. 1.2, b).



(a) Linear (b) RBF

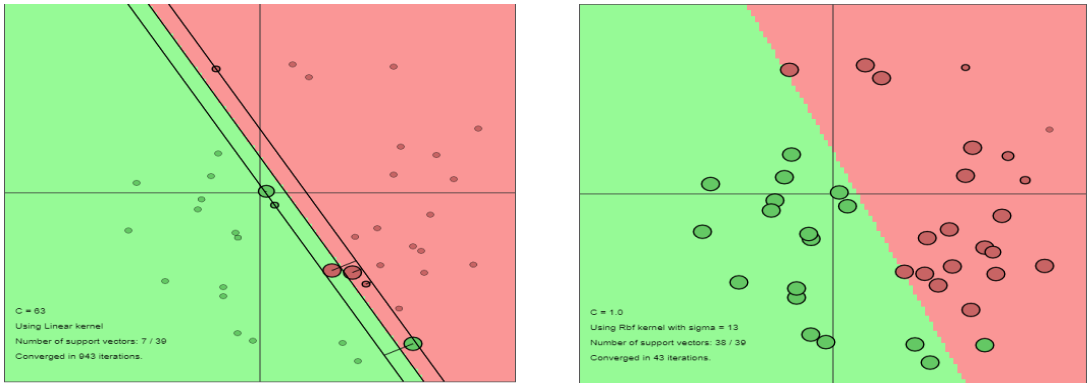
Figure 1.2: Stanford demo

The parameters C and σ substantially impact the behaviour of an SVM. Note that σ only applicable in the case of an RBF kernel.

- C is responsible for parametrizing the regularization of the slack variables. For the linear kernel, it impacts the width of the margin. That is, the higher the penalization of misclassification, the narrower it becomes and vice versa (Fig 1.3, a). As a result, this also changes the amount of support vectors that constitute the solution. Moreover, the additional box constraints ($0 \leq k \leq C$) in the non-separable case also restrict the number of support vectors and hence the degree of sparsity.

- Although not applicable to the linear kernel, σ regulates the kernel bandwidth of the Gaussian radial basis function. As such, it scales the squared Euclidean distance within the exponentiation (1.1). On one hand, very small values of σ rescale the distance in the original space such that the distance in the mapped feature space is inflated, resulting in a highly nonlinear decision surface. On the other hand, very large values of σ make the kernel evaluate to approximately 1. Indeed, for $\sigma \to \infty$ the fraction within the exponentiation will tend towards zero, ultimately yielding $\exp(0) = 1$. It can be verified that this results in a linear decision surface, as illustrated in Fig 1.3, b.

$$K\left(x, x_k\right) = \exp\left(\frac{-\left|x - x_x\right|\|_z^2}{\sigma^2}\right) \tag{1.1}$$



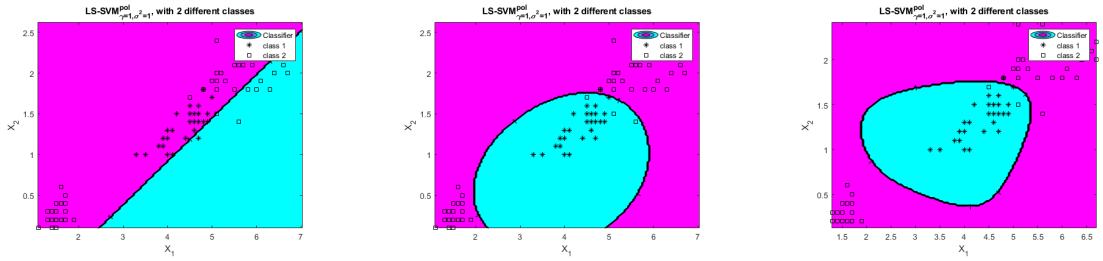
(a) Linear (b) RBF

Figure 1.3: Stanford demo (2)

1.1.3 LS-SVM Classifier

Hyperparameters and kernel parameters

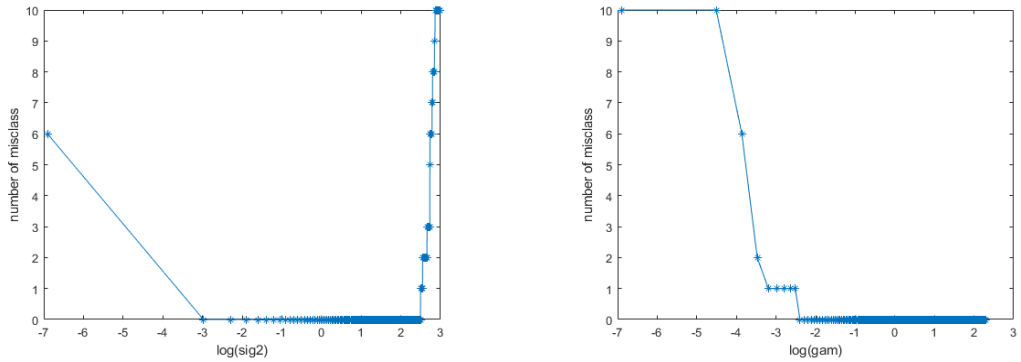
The first objective is to fit an LS-SVM with a polynomial kernel of varying degrees. A first degree polynomial (Fig. 1.4, a) kernel resulted in quite a poor test performance, 55% misclassifications to be specific. However, there were no more misclassifications for a polynomial kernel of degree $d = 3$ (Fig. 1.4, b). Since results are now satisfactory, further increasing the degree will likely lead to overfitting at some point (Fig. 1.4, c).



(a) $d = 1$ (b) $d = 3$ (c) $d = 5$

Figure 1.4: Polynomial kernel with varying degrees

Next it is of interest to evaluate a reasonable bandwidth for the RBF kernel w.r.t. the iris dataset. It would seem reasonable to choose σ^2 anywhere between $e^{-3} = 0.05$ and $e^2 = 7.39$ (Fig. 1.5, a). Past this point, the amount of misclassifications quickly starts to increase back again due to overfitting. A good range for the γ parameter on the other hand seems to start at $e^{-2} = 0.14$ (Fig. 1.5, right). Within the range of attempted values, classification performance does not seem to deteriorate for very large values, unlike what was observed for σ^2.



(a) σ^2 (b) γ

Figure 1.5: Brute-fore parameter tuning

Validation

Although the previously obtained parameters lead to good performance, it is not good practice to evaluate parameters w.r.t to a test sample. Instead, is is better to adopt a validation approach. To that end, the training sample is split into training and validation parts, which can be done in varying degrees of granularity (Fig. 1.6). Logically, with a finer grid one can more precisely identify the appropriate parameter, yet this obviously comes at a much higher computational cost. Even a single random split can already give a good idea on what range one should consider. With this in mind, it is essentially a

trade-off between computational demands and specificity of the parameter choice. As a compromise, k-fold cross-validation is often a reasonable choice.

For this example in particular, σ^2 seems to be most influential. As can be seen in Fig. 1.6, a lower value is recommendable. γ on the other hand, seems to have quite little influence once a good value of σ^2 is obtained.

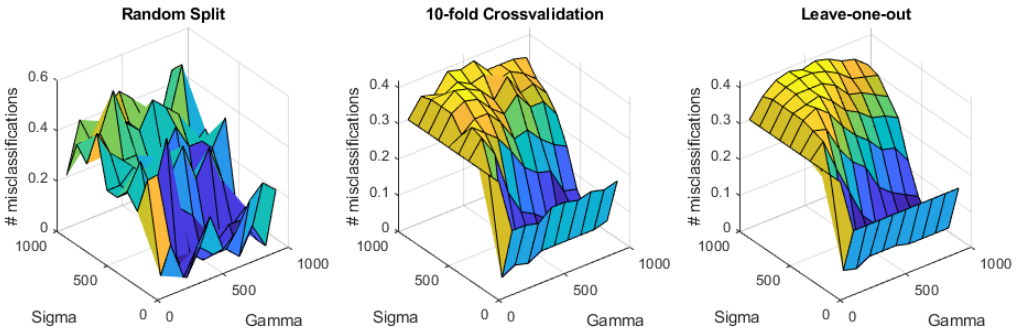


Figure 1.6: Comparison of validation approaches

Automatic tuning

As opposed to manually setting parameters after cross-validation, it is also possible to rely on automatic procedures such as the simplex algorithm or exhaustive grid search. Simplex optimization is an iterative procedure that continues until convergence, whereas grid-search actually evaluates all the values on a predefined grid. As such, grid-search provides more stability in the parameter it decides on, whereas the tuned parameters vary quite substantially when using the simplex method. On a different note, simplex tends to be less computationally demanding in comparison to brute force methods such as grid search. As such, grid search is much less feasible for larger problems. Note that there are many parameter combinations that lead to very similar cost function outcome (Fig. 1.6). Consequently, it is not surprising that the simplex solutions differ substantially over several runs.

ROC curves

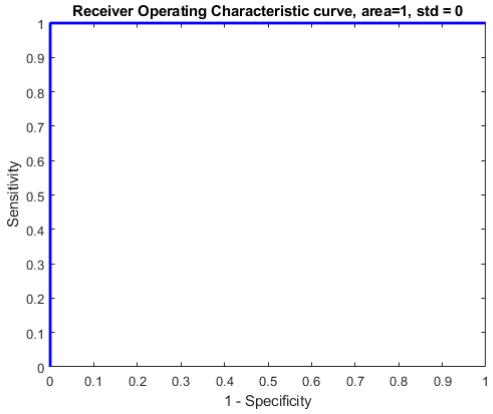


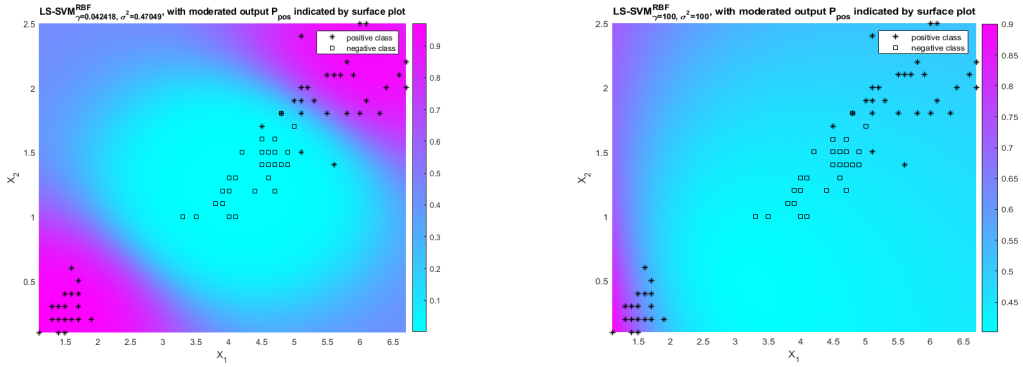
Figure 1.7: ROC curve for iris test set

The purpose of evaluating the ROC for the test set is to prevent simply optimizing a model on a given training set, which is quite useless with generalization in mind. In order

to develop a model that performs well on unseen data, it is therefore better to evaluate the test set instead. For the iris data in particular, tuned parameters are used to train the model and predict the test set instances. Fortunately, the performance that was obtained during training is carried over to the test set (Fig. 1.7). As the area under the curve (AUC) is 1, all test instances are correctly classified. More specifically, all actual positives are classified as such, yielding a sensitivity of 1. Furthermore, all actual negatives are also predicted to be negative, yielding a specificity of 1 and thus 1 - specificity = 0.

Probability estimates

Predictions do not provide any information on on the probability to belong to a certain class. By application of the Bayesian framework however, such estimates can be obtained (Fig. 1.8).



(a) Optimal parameters (b) sub-optimal parameters

Figure 1.8: Bayesian class probabilities

The higher the probability in a certain region, the more likely it is that an observation in that region belongs to the positive class and vice versa. As can be seen, this is quite informative for optimized parameters. In contrast, using inappropriate parameters causes the estimates to be less informative, which is not surprising since they are obtained using a sub-optimal model.

1.2 Homework Problems

1.2.1 Ripley

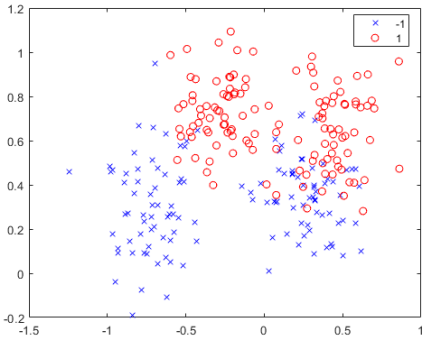
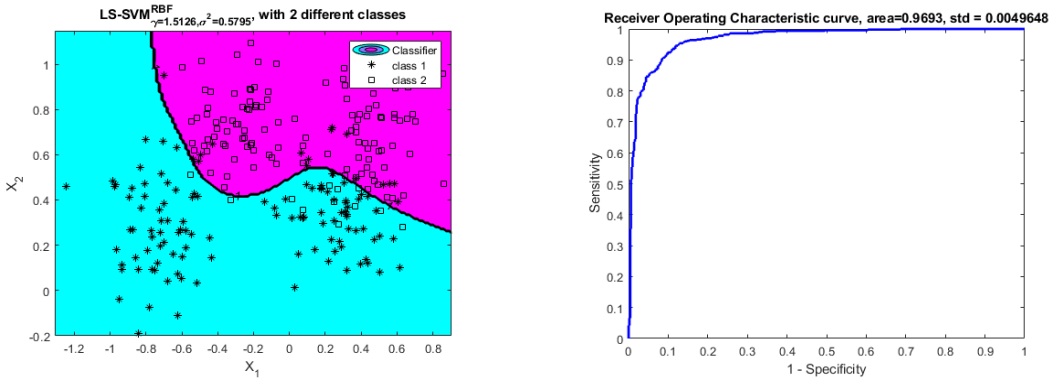


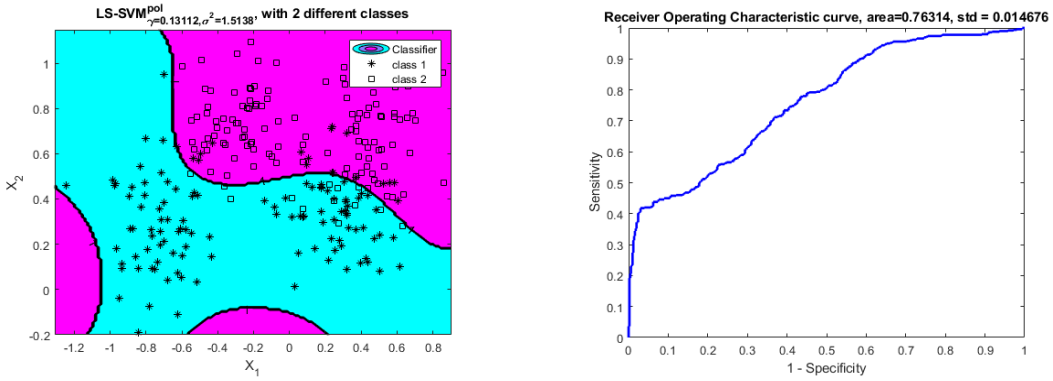
Figure 1.9: Ripley data

The low dimensionality of the ripley data makes it convenient to obtain a simple scatter plot of the data (Fig. 1.9). As can be seen, a linear model will underfit the problem, motivating the use of a non-linear model. To obtain an adequate model, the associated parameters are optimized using grid-search with 10-fold cross-validation, which is feasible since the data only contain a small amount of samples. Using this approach, the results for the RBF kernel (Fig. 1.10) are significantly better than those obtained with an optimized polynomial model (Fig. 1.11). Indeed, the RBF kernel results has solid performance on the test set.



(a) Classification (b) ROC

Figure 1.10: RBF model for ripley data



(a) Classification (b) ROC

Figure 1.11: Polynomial model for ripley data

1.2.2 Breast Cancer

Unlike the ripley data; the dimensionality of the input space (30) is much higher for this data set. As such, standard visualization techniques are not evident. In order to still get an idea about the underlying structure of the data, I have used a dimensionality reduction technique known as t-SNE (Fig. 1.12). This is a distance based method that tries to represent high dimensional data in terms of lower dimensional embeddings such that (dis)similarities can be visually represented. When applied to the wisconsin breast cancer data, some preliminary insights can be obtained. Firstly, the input space must contain valuable information with regards to the outcome variable since there are two distinct clusters. However, note that the shape and scale in which they are represented are not necessarily indicative of the actual topology in higher dimensional space.

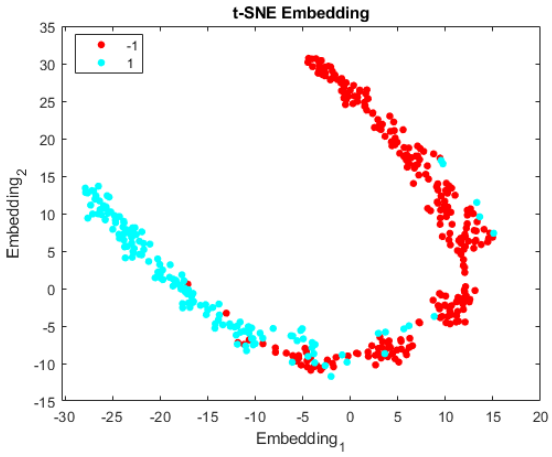
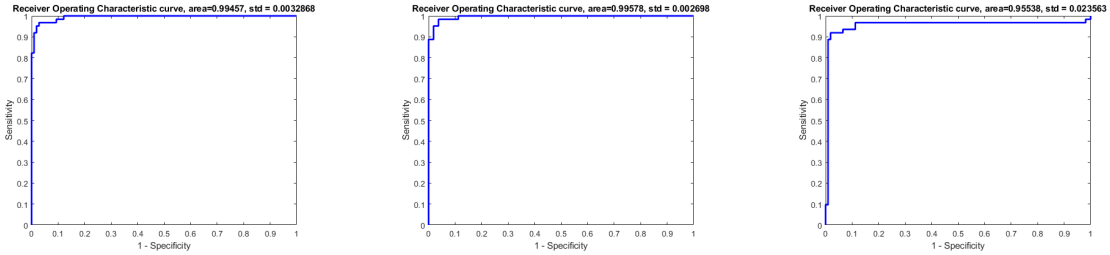


Figure 1.12: 2D t-SNE embedding

Once more, the appropriate parameters are identified by means of exhaustive grid-search with 10-fold cross-validation. However, the visualization does not give a good indication of what kernel to use for modelling. As such, several kernels are evaluated, namely linear, RBF and polynomial. As it turns out, both the linear kernel and the RBF kernel seem to perform extraordinarily well on this data set (Fig. 1.13). Although the polynomial model is able to attain a respectable AUC as well, it is still suboptimal compared to the other two. With the simplicity of a linear model in mind, it is preferable to opt for the linear kernel in this case. Nonetheless, similar results could be obtained with an RBF kernel, yet at the cost of a more complex model, i.e. additional parameter.



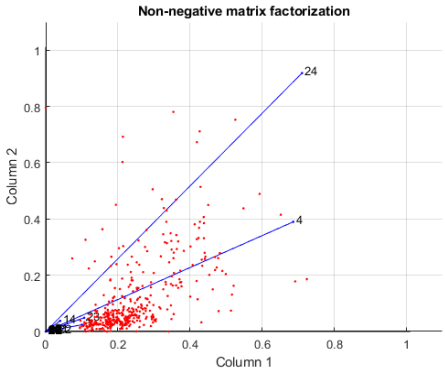
(a) RBF (b) linear (c) polynomial

Figure 1.13: Wisconsin: model performances

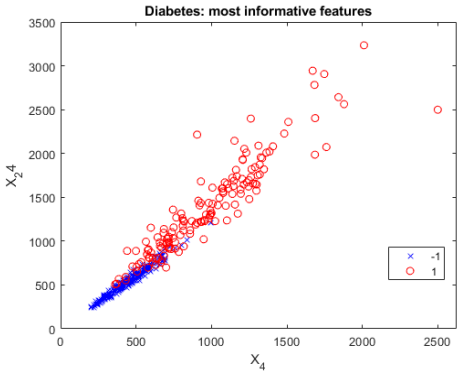
1.2.3 Diabetes data

Once again, visualization is troublesome because of the high dimensionality of the data. However, given the all the dimensions are non-negative, this data set is suitable for non-negative matrix factorization. That is, the input matix can be approximated by factorizing it into two non-negative matrices with a lower dimensionality, 2 in this case. As can be seen in Fig 1.12, variables 4 and 24 provide relatively strong weights for both columns of the deconstructed input space.

Not only has the matrix factorization allowed to visualize the data, it has also provided a way to select the most important features. As can be seen in Fig. 1.12b, the data are very clearly structured by these two variables alone. Consequently, only these variables will be used in the subsequent modelling process. In the end, performance results were almost identical compared to models using all 30 variables.



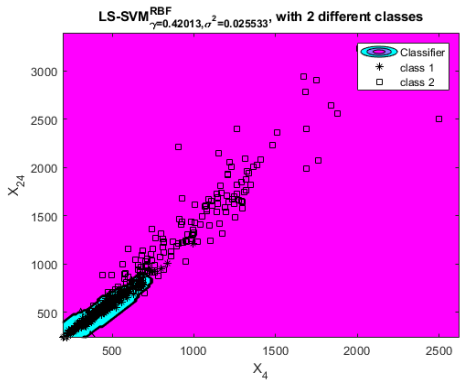
(a) NNMF



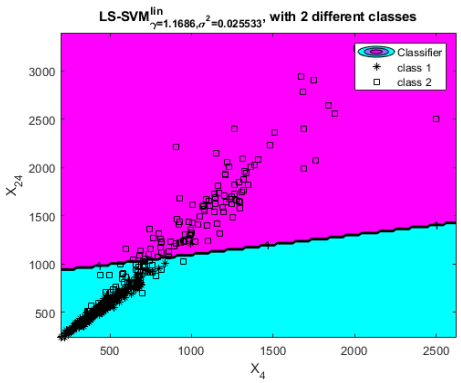
(b) Scatter plot

Figure 1.14: Visualization of diabetes data

Although the RBF kernel resulted in the best test performance (98%), it seems that is uses a very sharp decision boundary that is rather close to overfitting the data (Fig. 1.15, a), even though optimized parameters were used in the modelling process. While only sustaining a very minor performance drop (97%), the linear model seems to be quite adapt at providing a good decision boundary (Fig. 1.15, b). Once again, I would opt for the linear kernel for reasons of model simplicity and reduced risk of overfitting.



(a) RBF decision boundary



(b) linear decision boundary

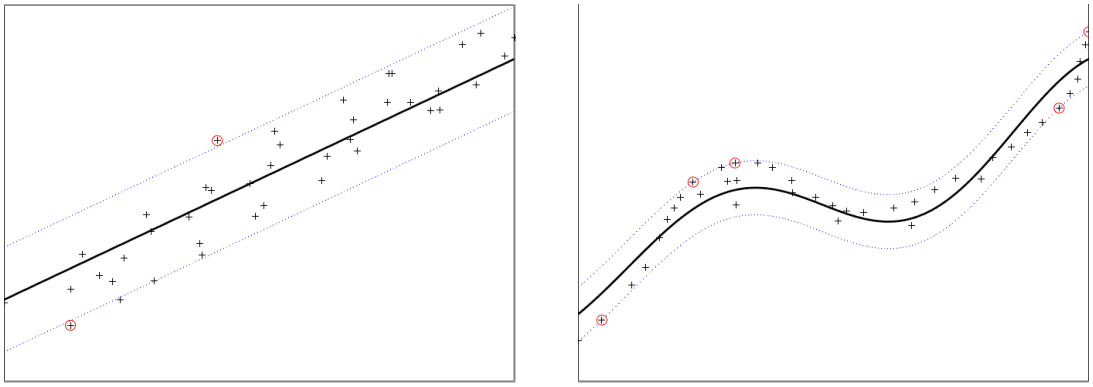
Figure 1.15: Model visualizations

Session 2: Function Estimation and Time Series Prediction

2.3 Exercises

2.3.1 SVM Regression

Firstly, a data set is constructed such that a linear model is most adequate (Fig. 2.1, a). A sparse linear regressor is obtained for a bound of 1 and epsilon 0.5. With these settings, almost all data points are contained within the ϵ-tube, except for the 2 support vectors indicated in red. Secondly, it is easy to construct a different data set that requires a non-linear model such as depicted in Fig. 2.1, b. In order to model these data, an RBF model with parameters $\sigma^2 = 0.5, \epsilon = 0.25$ is used with a bound of 1. Once again, this model specification leads to a sparse solution since only 5 support vectors are needed to obtain the regression line and corresponding tube.



(a) linear (b) RBF

Figure 2.1: SVM regression

SVM regression is different from standard LS regression in several ways. With regards to the loss function, standard SVM regression makes use of an L1-loss whereas least-squares naturally makes use of an L2 loss function. Moreover, the loss is adapted such that there is an insensitivity for values that deviate less than ϵ from the regression line. As such, only values outside the tube are taken into account in the loss function. With standard LS, all data contribute to the L2-loss. On a different note, the LS estimates follow directly from an unconstrained optimization problem, whereas the SVM regressor follows from a constrained optimization problem such that (most) of the data is located within the margin. Consequently, such a margin is obviously not present in LS.

2.3.2 LS-SVM Regression

In this next example, a sinc function is approximated using an LS-SVM model with RBF kernel. Upon evaluation of several (γ, σ^2) parameter combinations, a minimum MSE was attained for $(\gamma, \sigma^2) = (1000000,\ 1)$ (Table 2.1).

Parameters (γ, σ^2)	MSE
(10, 0.01)	0.009
(10, 1)	0.032
(10, 100)	0.134
(1000, 0.01)	0.009
(1000, 1)	0.0105
(1000, 100)	0.117
(1000000, 0.01)	0.0130
$(\mathbf{1000000}, \mathbf{1})$	0.008
(1000000, 100)	0.111

Table 2.1: Parameter combinations and MSE

As it turns out, these parameters lead to a model that also performs well on the test data (Fig. 2.2). Even though these might not be the optimal parameters yet, they still result in a quite satisfactory performance. The idea of finding the optimal parameter is somewhat idealistic with this approach since, for real-world applications, the underlying function is unknown.

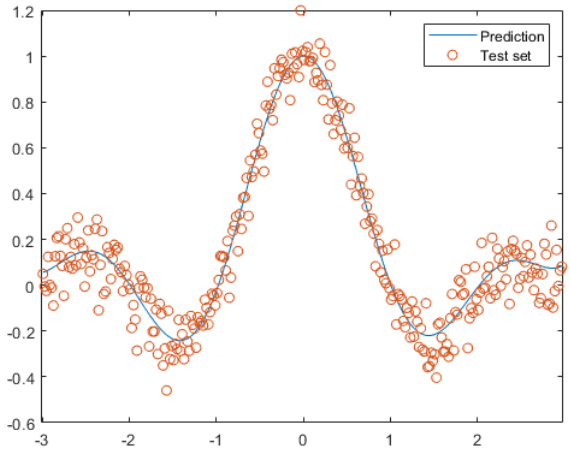


Figure 2.2: Test set predictions

With automatic tuning on the other hand, different parameters are obtained:

- Simplex: $\{\gamma = 1427,\ \sigma^2 = 0.361\}$
- Grid-Search: $\{\gamma = 4601,\ \sigma^2 = 0.487\}$

Results tended to vary quite substantially across runs. Note therefore that these reported parameter results are averages over 10 runs for each of the algorithms. In any case, these are quite different from the parameters that were obtained in the previously adopted approach.

Bayesian Framework

The Bayesian framework involves 3 different levels in inference. At each of these levels, the model (hyper)parameters follow from the maximum posterior solution to Bayes' rule, which is obtained by combining information from the data and prior knowledge. To ensure one is working with a probability density, a normalizing constant, or evidence, is used such that the sum of all probabilities sums to 1. More specifically, this term takes the form of a marginal likelihood which does not depend on the parameters on is trying to find the maximum posterior solution for. Consequently, it can be used as an actual likelihood for the remaining parameters and is used as such in the subsequent level of inference.

- Level 1: model parameters w, b follow from the maximum posterior solution w_{MP}, b_{MP}
- Level 2: hyperparameters μ, ζ respectively regulate the influence of the model parameters and the error variables and also follow from maximizing the (log) posterior. Moreover, they determine the effective number of parameters of the model. If the number of parameters is substantially less than obtained with the regular LS-SVM model, one has a sparser model.
- Level 3: lastly, kernel-specific parameters can be tuned by comparing model quality in terms of Bayes factor.

Using this approach, the sinc function is revisited using an RBF kernel. After optimizing the (hyper)parameters and kernel parameter σ^2 at their corresponding levels of inference, a Bayesian LS-SVM model is obtained. As can be seen in Fig. 2.3, the optimization approach leads to a very accurate model. What's more, confidence bands can also be provided since we are working in a probabilistic setting now. As such, Bayesian LS-SVM offers an appealing alternative to regular SVM regression in that it still provides an ϵ-tube of sorts.

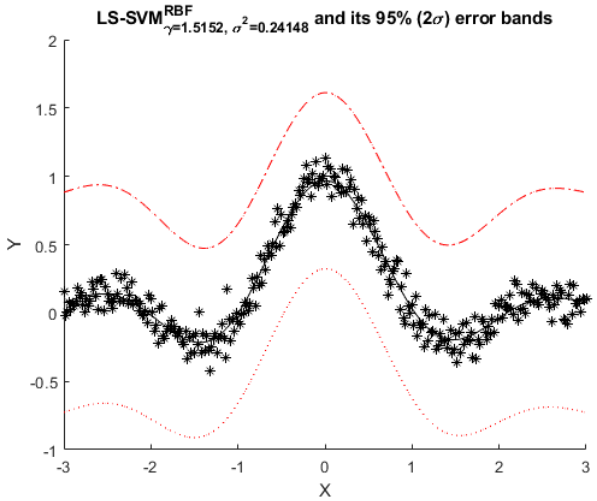


Figure 2.3: Bayesian LS-SVM regression with confidence bands

Automatic Relevance Determination

Within the Bayesian framework, one can also identify important variables within a given problem. More specifically, the kernel matrix is adapted by introducing a weighting matrix that will ultimately reflect the importance of the corresponding elements. Since this involves the kernel matrix, optimization can be conducted within the third level of inference. In the case of the sinc function, this mechanism is well demonstrated (Fig. 2.4). Indeed, it can correctly separate function patterns (X_1) from noise (X_2, X_3).

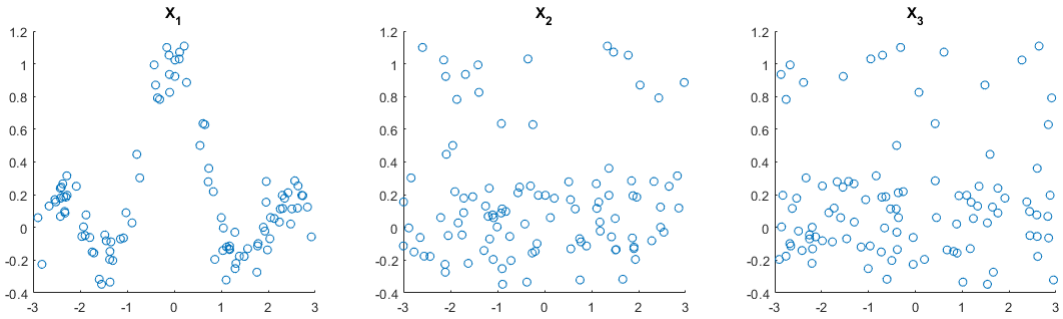
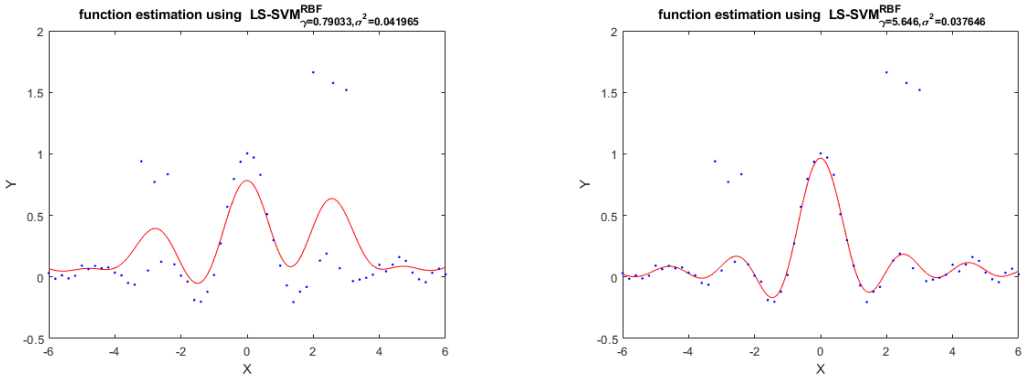


Figure 2.4: Automatic relevance determination

Robust LS-SVM

In this section, it is of interest to obtain models that are resistant to the influence of outliers. To that end, the MSE (L2) cost function is replaced by the MAE (L1) cost function. Using an L1 loss allows to suppress the impact of strong deviations since the absolute value is taken as opposed to squaring them. Moreover, the error variables are modified by introducing weights such that their influence is controlled by the robust estimate of their standard deviation (IQR). As such, a weighted LS-SVM model is ultimately obtained. This mechanism is well demonstrated in Fig. 2.5.

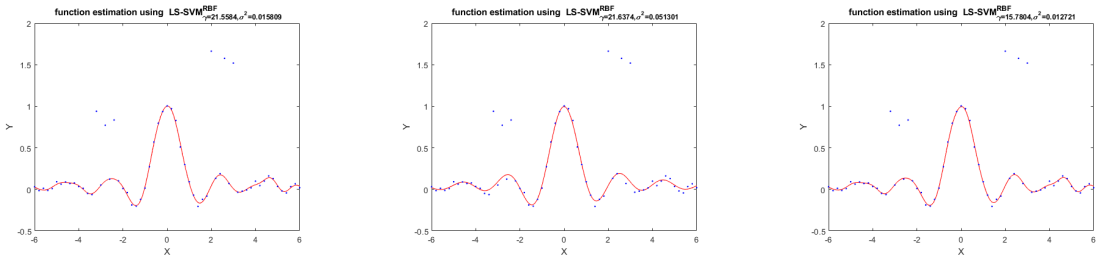


(a) Non-robust (b) Robust (huber)

Figure 2.5: Robust regression

While the standard model (a) has become inaccurate due to overly emphasized the outliers, the robust model (b) is able to maintain an accurate estimate of the actual function by suppressing the influence of these data points. Note that the loss function was chosen to be the Huber loss. For large deviations, it penalizes the errors by means of an L1 estimator, whereas more reasonable deviations are penalized by an L2 estimator.

With regards to the weighting scheme of the error variables, there are several options available, each exhibiting slightly different behaviour in the way they deal with deviations (Fig. 2.6). The Hampel loss (a) is similar to the Huber loss, yet it is more stringent for very large deviations since it simply disregards them past a certain threshold. On the other hand, loss functions such as the logistic loss (b) or the myriad loss (d) penalize the deviations inversely proportional to their size.



(a) Hampel (b) Logistic (c) Myriad

Figure 2.6: Loss functions

2.4 Homework Problems

This last sections covers two time-series problems to be solved by means of a tuned LS-SVM model.

2.4.1 Logmap

The logmap data contains a sequence of 150 training observations and another sequence of 50 test set observations. In order to obtain model inputs that allow for recurrent prediction, a rolling window with a stride of 1 is constructed for a certain order, or lag, on the training sequence. The order thus takes the form of an additional hyperparameter that can equally well be tuned. To that end, several order, e.g. 10,20,30,40,50 are considered. For each of these input structures, the RBF parameters (γ, σ^2) can be tuned accordingly. By storing the MSE for each model, one can identify the preferred model specification, i.e. lowest MSE. Following this approach, a reasonably accurate model is obtained for the logmap data (Fig. 2.7). This parameters that constitute this model are {order=10, $\gamma = 66441, \sigma^2 = 441$}.

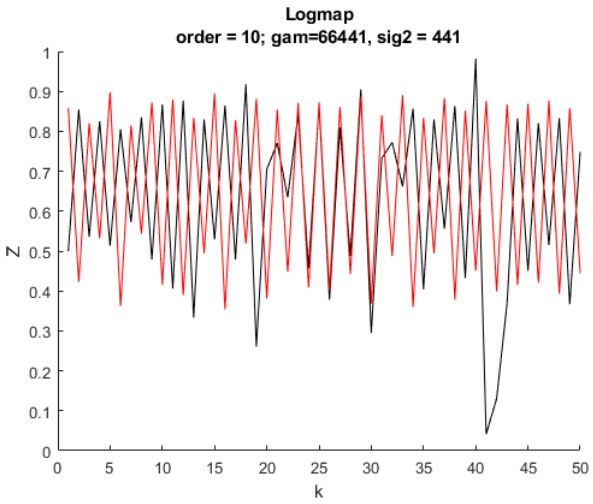


Figure 2.7: Logmap test set and predictions

2.4.2 Santa Fe

Although the Santa Fe test set contains more samples than the logmap test set, a lag of 50 might still be a reasonable input order. As it turned out, an order of 50 even resulted in a highly accurate model that is able to almost perfectly predict the dynamic of the time series (Fig. 2.8).

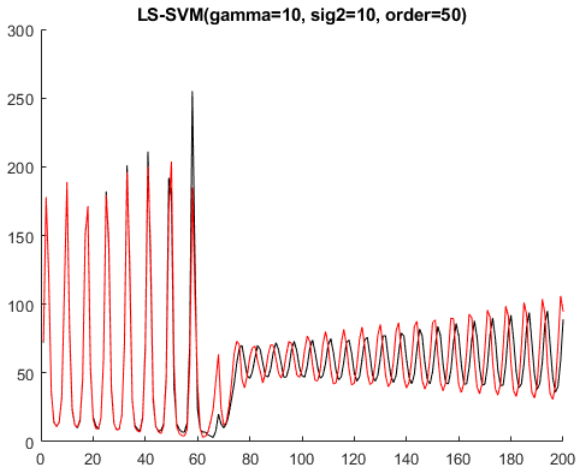


Figure 2.8: Test set predictions using suggested parameters

The previously adopted settings, together with the suggested order of 50, have already resulted in very well performing model. However, parameter optimization should still be conducted to verify in any improvements can be made to the model. The parameters resulting from the parameter optimization are $\{\gamma = 1351.0934, \sigma^2 = 10.1071, \text{order} = 30\}$. Although the γ value is substantially larger than before, σ^2 and order are not too far of from the initial modelling attempt. In any case, the performance improvements on the test set seem to be only very marginal, if any.

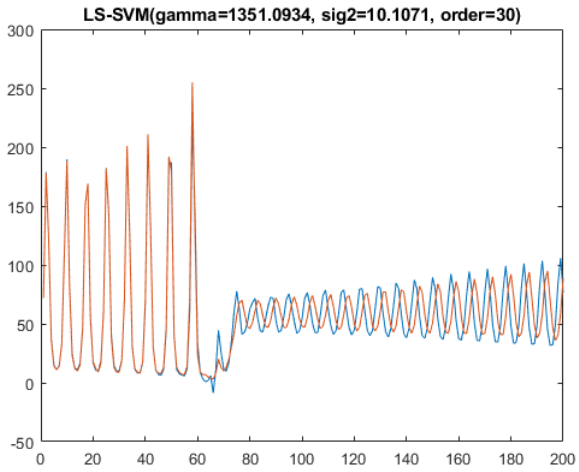


Figure 2.9: Test set predictions using a model with optimized parameters

Note that these parameters were obtained on a separate validation part of the training data. As usual, this is a more appropriate approach for such endeavours. Furthermore, there is an additional advantage in time-series settings. Predictions for the test set are obtained in a recurrent manner as follows

$$\hat{z}_t = f\left(z_{t-1}, z_{t-2}, \ldots, z_{t-n}\right)$$

If a validation set would not be held out, predicting test set observations would involve the use of preceding training observations. By withholding the last part of the training set (at least as much as the order of the model), one is able to circumvent this issue.

Session 3: Unsupervised Learning and Large-scale Problems

3.5 Exercises

3.5.1 Kernel PCA

Denoising

When using PCA, denoising can be achieved by deconstructing a feature space x into its principal components z and subsequently using the score variables to obtain a reconstruction of the original space $\tilde{x}$, (partly) removing noise in the process. This approach is quite nicely summarized by what is known as an information bottleneck (Fig. 3.10).

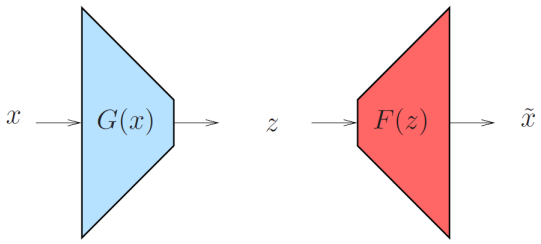


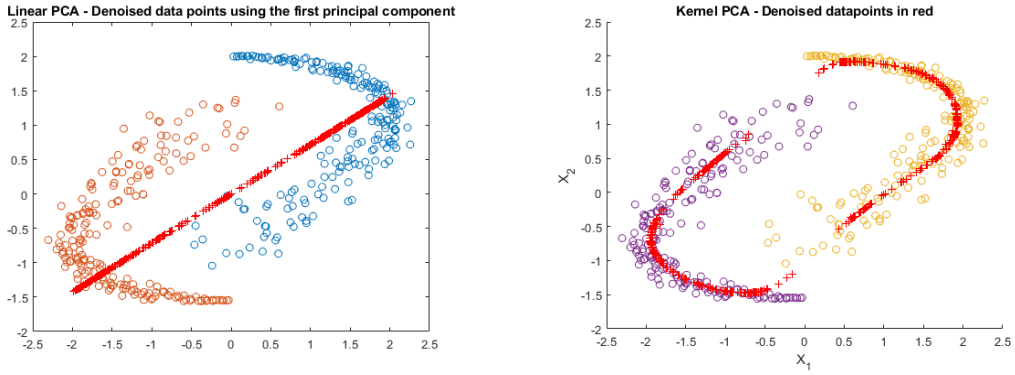
Figure 3.10: Information bottleneck

Ideally, one aims to minimize a reconstruction error (3.2) with as few score variables as possible. Depending on the desired degree of denoising, the number of principal components can be increased as to obtain a more accurate reconstruction of the original space. Logically, the value of adding more components decreases marginally since the sorted eigenvalues are ever decreasing in terms of contribution.

$$\min \sum_{k=1}^{N} \|x_k - \tilde{x}_k\|_2^2 \tag{3.2}$$

Linear Vs Kernel PCA

On one hand, linear PCA finds the projected variables with maximal variance in the actual input space. As such, the decomposition contains at most the amount of variables in that same input space. On the other hand, Kernel PCA implements the same mechanism in the feature space. More specifically, it identifies the directions of maximal variance in a higher dimensional space after mapping the inputs to that space. As such, the solution can consist of more components than what constitutes the input space. Depending on the structure of the data, one should choose the most appropriate approach. For the yin-yang data with added noise (Fig. 3.11), a non-linear method such as kernel PCA is evidently more suitable.



(a) linear (b) Kernel

Figure 3.11: Denoising a noisy yin-yang distribution

Tuning the number of components

Tuning the number of components can not simply be done based on the reconstruction error since, for each additional component, the reconstruction error obviously decreases. Instead, it is more sensible to tune the γ parameter of our LS-SVM model

$$\begin{aligned} \max_{w,e} \quad & \gamma \frac{1}{2} \sum_{k=1}^{N} e_k^2 - \frac{1}{2} w^T w \\ \text{subject to } & e_k = w^T \left(\varphi \left(x_k \right) - \hat{\mu}_\varphi \right), k = 1, \ldots, N \end{aligned}$$

More specifically, γ is introduced into the eigenvalue problem by setting $\gamma = 1/\lambda$. As such, the number of components can be tuned indirectly by tuning γ. As such, λ acts as s threshold such that each component must have a corresponding λ larger than that threshold. Lastly, tuning the kernel parameters such as σ^2 for RBF, can best be done by holding out a validation set. Alternatively, new representative points could also be generated for this particular yin-yang data structure.

3.5.2 Spectral Clustering

Whereas common clustering techniques such as K-means use a distance based approach to obtain a clustering solution, spectral clustering instead relies on the connectivity among nodes of a graph, regardless of the distance between the points. In order to do so, the graph $G = (V, E)$ is represented by an adjacency matrix containing the similarities between the vertices $V = \{v_1, \ldots, v_n\}$. Towards obtaining the clustering solution the space can be partitioned by taking the minimal cut

$$\min_{q_i \in \{-1,+1\}} \frac{1}{2} \sum_{i,j} w_{ij} \left(q_i - q_j \right)^2$$

which can be approximated by conducting an eigen decomposition of the graph Laplacian matrix for G, yielding eigenvalues $\{\lambda_1, \ldots, \lambda_n\}$ and eigenvectors $\{v_1, \ldots, v_n\}$. From this solution, the second largest eigenvalue λ_2 indicates how tightly connected the nodes are in the graph. Note that for G to be a connected graph in the first place, one should have $\lambda_1 = 0$. In the case of a bipartite problem, clusters are obtained by assigning the elements of eigenvector v_2 to their corresponding nodes and grouping the nodes with positive and negative eigenvectors into two separate clusters. For k clusters, one has to consider k eigenvectors of the normalized Laplacian instead, yielding a matrix in which every node

is represented by its corresponding row. To now obtain the clustering solution, K-means can be applied on this eigenspace embedding.

In usual clustering settings, one is faced with an unsupervised problem since the points are unlabeled. In this case, the difference between clustering and classification is most obvious as it takes the form of an unsupervised and a supervised problem respectively. However, for spectral clustering, there are also some similarities with classification. For instance, the procedure to find the minimal cut involves a binary decision for determining which set a particular point will belong to given a certain cut. In that sense, binary spectral clustering is more related to classification than classical algorithms such as K-means since they are completely distance based.

To demonstrate the power of the kernel variant of spectral clustering, two interlaced rings in 3D-space are considered (Fig. 3.12). Clearly, standard clustering techniques would struggle to deliver a satisfactory solution.

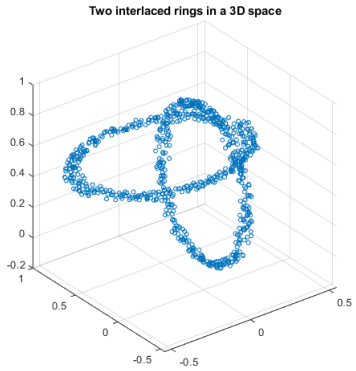
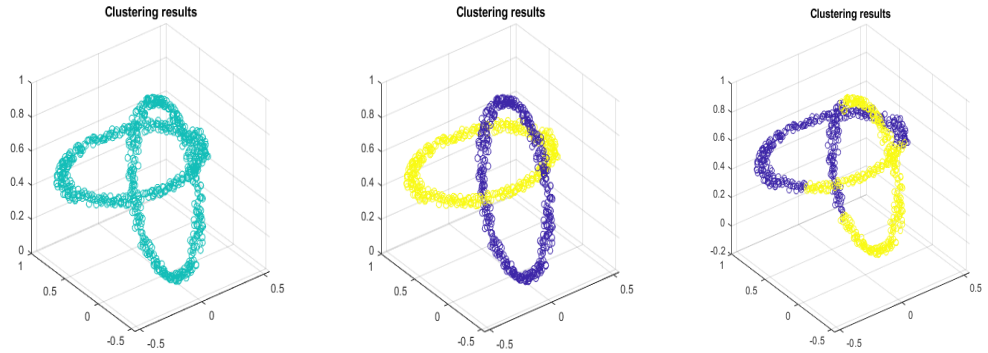


Figure 3.12: Interlaced rings in 3D space

With the appropriately specified σ^2, one can obtain an excellent clustering solution. For this example in particular $\sigma^2 = 0.01$ seems to yield the best solution (Fig. 3.13). Similar to the affinity matrix of the undirected graph in regular spectral clustering, the kernel matrix is used a similarity measure in kernel spectral clustering. As such, it is evident that one should adopt an appropriately parametrized kernel function, such that the subsequent eigenvalue decomposition is carried out on information that accurately represents the pairwise distances in the original dataa.



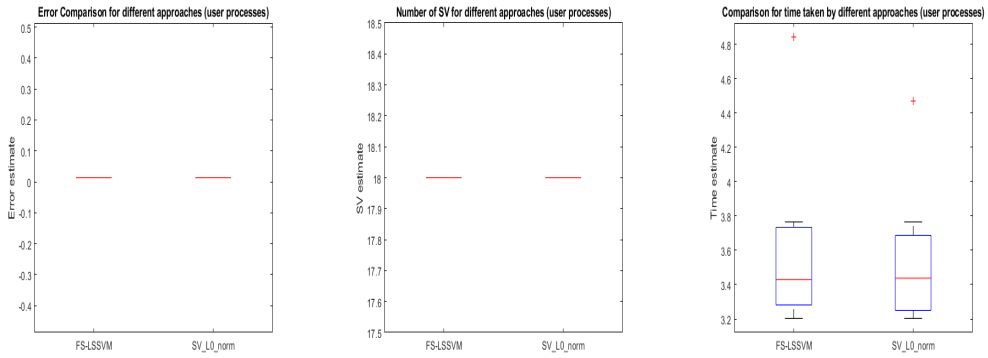
(a) $\sigma^2 = 0.001$ (b) $\sigma^2 = 0.01$ (c) $\sigma^2 = 0.1$

Figure 3.13: Clustering results for different σ^2

3.5.3 Fixed-size LS-SVM

In large scale settings, obtaining a solution in an efficient and timely manner is still very much a challenging problem. For the dual formulation of the model, the solution in the α multipliers depends on the size of the kernel matrix which quickly grows as the number of training data n grow, i.e. $n \times n$. On the other hand, the solution does not depend on the dimensionality of the input space in the dual formulation, making (LS)-SVM particularly useful for high-dimensional problems on small to moderately sized data sets. In contrast, the solution to the primal formulation takes the form of the w vector, hence depending on the dimensionality of the input space rather than on the amount of data. Consequently, the primal formulation leads to a more efficient computation of the solution in settings that involve a lot of data. With these considerations in mind, the appropriate formulation should be based on the nature of the problem at hand. However, the primal solution is not computable for a non-linear (unknown) feature map that is possibly infinite in dimension. In such cases, a finite dimensional approximation of the feature map is necessary to be able to solve the primal problem, as is done in fixed-size LS-SVM.

To obtain such an approximation, fixed-size LS-SVM implements the Nyström method for a working set of M support vectors. However, unlike the original Nytström method, the subset selection for fixed-size LS-SVM is not conducted at random. Instead, the support vectors are selected according to the quadratic Renyi entropy, which is known to act as an index of diversity. As such, it the resulting subset is likely to be a sufficiently diverse representation of the entire data set. Of course, this diversity is only indicative of the distribution in the dual space as the computed entropy involves the small kernel matrix. For kernels such as RBF, this does imply the bandwidth σ^2 to influence the diversity among the data in the dual space.



(a) errors (b) number of SV's (c) training time

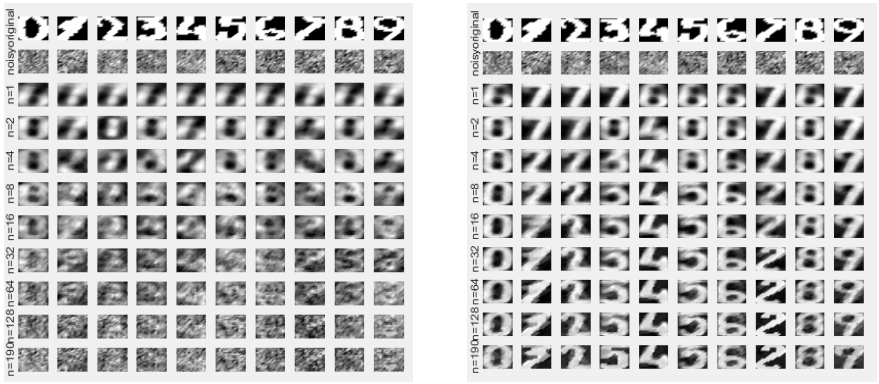
Figure 3.14: FS-LSSVM versus ℓ_0-approximation

Empirically, the results for regular fixed-size LS-SVM are compared to those obtained after using an ℓ_0-approximation. This approximation can, in addition to efficiently solving large scale (non-linear) LS-SVM problems, also result in a sparse solution. As can be seen in Fig. 3.14, the solution leads to an exactly equal amount of support vector (b). This is somewhat surprising since the ℓ_0-approximation has not resulted in any sparsity whatsoever. As such, it is evident that the error difference is negligible. The computation time for both methods is also very similar. Perhaps a different data set would have better been able to demonstrate the supposed sparseness.

3.6 Homework Problems

3.6.1 Kernel PCA

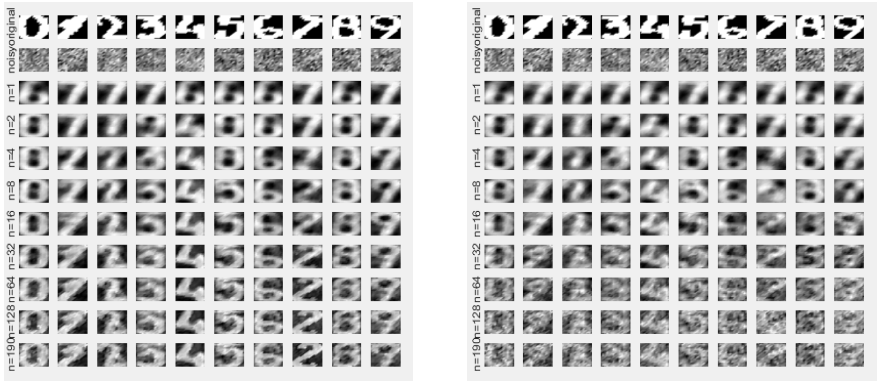
For this first homework problem, the objective is to denoise digit images using kernel PCA. For comparative purposes, linear PCA has been to do so as well. Based on the results of these processes, kernel PCA (Fig. 3.15, b) clearly outperforms linear PCA (Fig. 3.15, a) in its ability to denoise while maintaining a sharp pixelated representation of the digit images, even for a relatively low amount of components. Note that the suggested noisefactor of 1 was adopted towards obtaining these results.



(a) Linear (b) Kernel

Figure 3.15: PCA for digits data

The σ^2 parameter was estimated using a rule of thumb provided in the script, rescaled by a factor of 0.7. This approach has seemed to yield quite satisfactory denoising results. To analyze the impact of this parameter on the, we could simply consider a different factor. For instance, a factor of 1.4 (Fig. 3.16, a) would already result in a σ^2 that substantially bigger than what is suggested by the rule of thumb. For illustrative purposes, an unreasonable factor of 7 is also considered (Fig. 3.16, b). Clearly, using overly inflated σ^2 values tends to worsen the quality of denoising results quite substantially. After all, a factor of 0.7 has turned out to be a reasonable correction of the rule of thumb, which is possibly already a good starting point of its own.



(a) $\sigma^2 = \sigma^2 * 1.4$ (b) $\sigma^2 = \sigma^2 * 7$

Figure 3.16: Impact of σ^2 on denoising results

3.6.2 Fixed-size LS-SVM

Shuttle (statlog)

The Shuttle data set contains 58,000 data points for 9 numerical attributes. Moreover, the data constitutes a multi-class problem of 7 classes in total, yet approximately 80% of the instances belong to class 1, making it quite an unbalanced class distribution. In the subsequent analysis, a subset of 700 observations is selected as to allow for more manageable computation times. The subset selection is done according to the optimization of the entropy criterion as provided in `fixedsize script1.m`. To now obtain a visualization of this part of the data set, I have adopted standard PCA as a dimensionality reduction technique such that the components and corresponding scores can be represented in the component space (Fig. 3.18).

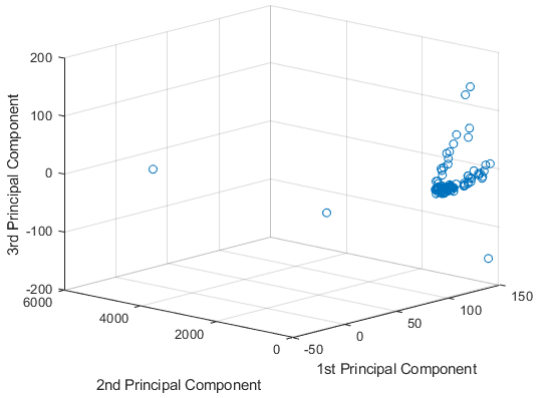
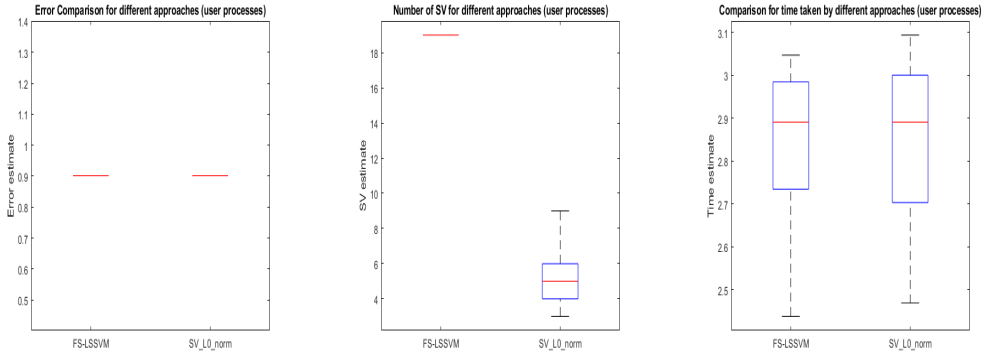


Figure 3.17: Shuttle Data in 3D principal component space

As can be seen, the visualization is mainly indicative of the importance of the first principal component accounting for approximately 94% of the variance. In turn, this component is highly correlated with attribute 6, as indicated by the 0.9943 loading with that attribute. Although the first principal component seems to give a good indication of an important variable, the relevance of the second component seems to be mainly caused by two anomaly points. Together with the third component, an additional 5% of the variance is accounted for.



(a) errors (b) number of SV's (c) training time

Figure 3.18: FS-LSSVM versus ℓ_0-approximation for Shuttle subset

By making use of an adjusted version of `fslssvm script.m`, fixed-size LS-SVM results are obtained (Fig. 3.19). More specifically, the fixed-size method attains an estimated error

of about 0.9 (a), using an estimated 19 support vectors (b). Although the ℓ_0-approximation once again attains the same error (a), it now does so with a substantially sparser model, only using about 4 to 6 estimated support vectors (b). Computation times are on a very similar scale (c).