Classification

James O'Reilly
james.oreilly@student.kuleuven.be

1 A Simple Gaussian Example

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

As we know the parameters of the underlying distributions from which the data were generated, we can use these parameters to classify new data generated from these distributions. Using the means of of the distributions for each class, a linear classifier can be made by bisecting the line connecting these two means. The generated data and classifier are given in Figure 1.

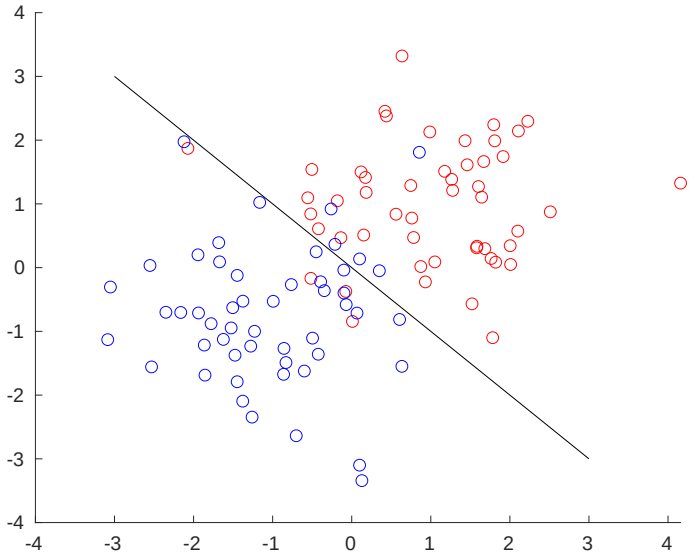


Figure 1: New data and linear classifier generated from underlying distributions.

This classifier is optimal in the sense of the Bayes decision rule. For a given class i and data x, Bayes rule gives that

$$P(C_i|x) \propto P(x|C_i)P(C_i) \tag{1}$$

In the case where $P(x|C_i)$ is normal, we have that the log of the conditional probability for a class C_i given by

$$\log \mathrm{P}\left(C_i \mid x\right) \propto -\frac{1}{2}\left(x-\mu_i\right)^T \Sigma_i^{-1}\left(x-\mu_i\right) - \frac{1}{2}\log\left|\Sigma_i\right| + \log \mathrm{P}\left(C_i\right) \tag{2}$$

for $i = 1, 2$. As $\Sigma_i = diag(1, 1)$ this becomes

$$\log \mathrm{P}\left(C_i \mid x\right) \propto -\frac{1}{2}\left\|x-\mu_i\right\|_2^2 + \log \mathrm{P}\left(C_i\right) \tag{3}$$

If the prior probabilities for each class are equal, then for class i we can use the decision rule

$$D_i = -\frac{1}{2}\|x - \mu_i\|_2^2 \tag{4}$$

Therefore, for a given point x, if $D_1 > D_2$, then x is in the first class, and is in the second class otherwise. This is then equivalent to the decision rule using the bisection of the line between the two class means.

2 Support Vector Machine Classifier

What is a support vector?
Mathematically speaking, support vectors are the data points $\mathbf{x}_i$ for which the Lagrange multipliers α_i are greater than zero. More specifically, consider the case of hard-margin SVMs where we want to find the solution to the minimisation problem. Therefore, depending on the different starting positions, different results are obtained by couple simulated annealing.

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 \tag{5}$$

In order to compute the bias b, the KKT conditions specify that the constraint for sample i must be tight. Hence, b depends only on those points for which $\alpha_i > 0$. Therefore, the solution to the minimisation problem depends only on all points $\mathbf{x}_i$ for which $\alpha_i > 0$.

In the case of soft-margin SVMS (or C-SVMs), the solution to the minimisation problem is given by

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_i \tag{6}$$

The KKT conditions specify that in order to compute the bias b, the constraints for $\alpha_i > 0$ and ξ_i must be tight. The solution therefore depends on the points for which $\alpha_i > 0$.

When does a particular datapoint become a support vector? A datapoint $\mathbf{x}_i$ becomes a support vector when the Lagrange multipliers α_i are greater than zero.

When does the importance of the support vector change? Illustrate visually.
The importance of a support vector can change because the parameters of the model are changed (see Figures 6), or because a new datapoint is added. Certain datapoints will not change the importance of support vectors. In the linear case, new datapoints that are outside classification margins and predict the correct class have little effect on importance. In the RBF case, new datapoints at the center of clusters have little effect on importance (see Figure 2 Datapoints that are at the inside the margin (in the linear case) or at the edge of a cluster (in the RBF case) will reduce the importance of similar points further from the boundary (see Figure 3).

What is the role of parameters C and sigma?
The C parameter determines the 'cost' of misclassifying each training example. For large values of C, the SVM optimisation will give a hyperplane with a smaller margin if that hyperplane has better classification. Conversely, a small value of C will cause the optimiser to look for a hyperplane with a larger margin, even if that hyperplane misclassifies more points. In this regard, C is essentially a regularisation parameter which controls the trade-off between achieving a low misclassification error on the training data maximising the width of the margin.

Structural risk minimisation (SRM) is partly implemented in SVMs by the tuning of the C parameter. The C parameter enforces an upper bound on the norm of the weights. If you look at the dual formulation for the optimisation problem for the SVM, there is a box constraint that prevents the dual parameters from exceeding C:

$$\max_{\alpha_k} \mathcal{Q}(\alpha) = -\frac{1}{2}\sum_{k,l=1}^{N} y_k y_l K\left(x_k, x_l\right)\alpha_k\alpha_l + \sum_{k=1}^{N}\alpha_k \tag{7}$$

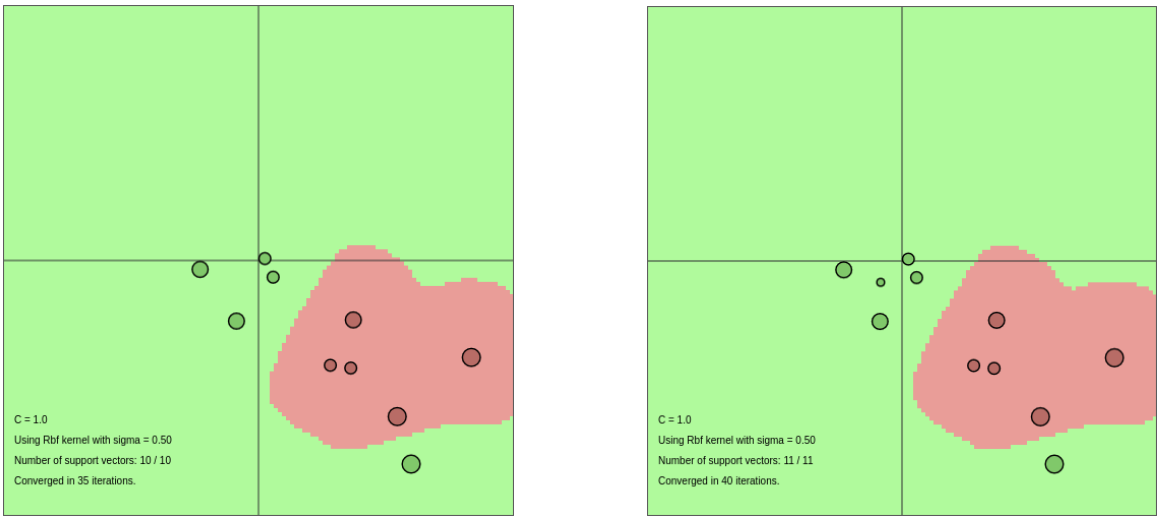


Figure 2: Adding a datapoint at the center of a cluster has little effect on importance of support vectors.

such that

$$\begin{cases} \sum_{k=1}^{N} \alpha_k y_k = 0 \\ 0 \leq \alpha_k \leq c, k = 1, \ldots, N \end{cases} \tag{8}$$

Imposing an upper bound on parameters in the dual formulation also imposes an upper bound on the norm of the primal weights. This means that C indexes a nested set of hypothesis classes, each determined by the upper bound on the norm of the weights. Therefore, as C increases, the complexity of the hypothesis class also increases. In this way, the C parameter implements structural risk minimisation by limiting the complexity of the hypothesis class.

The sigma parameter determines the region of influence of each datapoint. If the value of sigma is low, then every datapoint will have a small region of influence. Conversely, high values of sigma mean that datapoints will have a large region of influence. This is clear from the equation for the Gaussian RBF kernel:

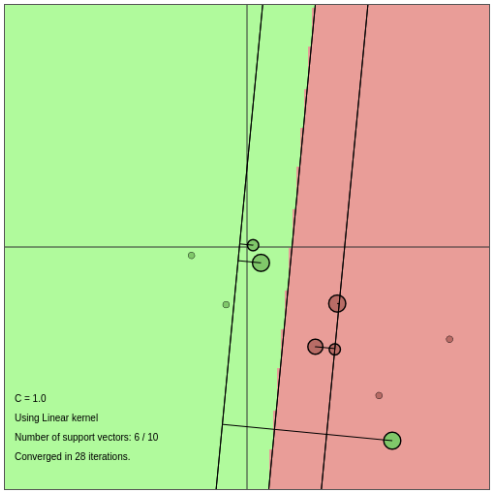
$$K(x_i, x_j) = \exp\left(-\|x_i - x_j\|_2^2 / \sigma^2\right) \tag{9}$$

With a low value of sigma, the SVM decision boundary will be dependent on only those points that are closest to the decision boundary, effectively ignoring points that are farther away. A high sigma value will give a decision boundary that considers points that are further from it. As a result of this, low values of sigma tend to give highly flexible decision boundaries, and high values of sigma result in more linear decision boundaries.

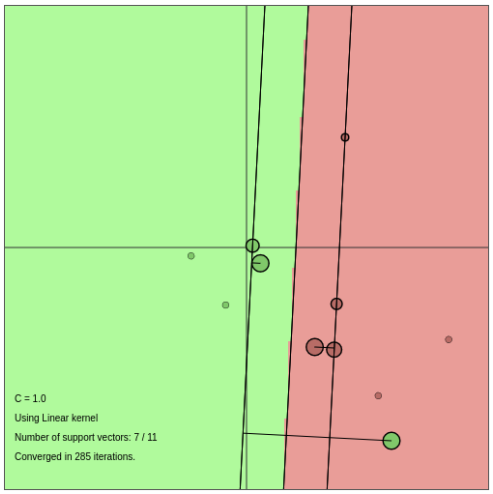
What happens to the classification boundary if you change these parameters. Illustrate visually.
In the case of the linear kernel, as C is increased, the margin must become smaller and in order to accommodate for this, the angle of the linear decision boundary changes so that the support vectors are closer to the boundary. This is very clear in Figures 4 and 5.

For the RBF kernel, as C is increased, the decision boundary becomes less flexible and more smooth (see Figure 6). As Sigma is increased, the range of effect of each data point is increased. This is clear in Figure 7, where for low values of sigma each point has a classification boundary, and for high values this boundary tends toward linear.

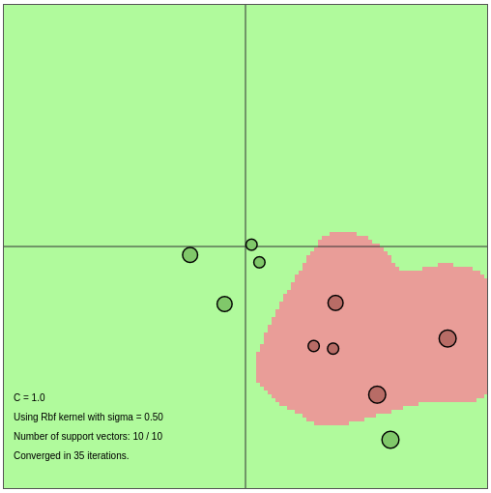
What happens to the classification boundary when sigma is taken very large? Why?
For the Gaussian RBF kernel, a large sigma means a large standard deviation of the Gaussian, and so the influence of each data point is larger. When sigma is very large, the model is too constrained and cannot capture the complexity or "shape" of the data. The region of influence of any selected support vector would include the whole training set. The resulting model will behave similarly to a linear model with a set of hyperplanes that separate the centres of high density of any pair of two classes. Keerthi et al wrote a paper titled *"The Asymptotic behaviour of Support Vector Machines with Gaussian Kernel"*, in which they discuss



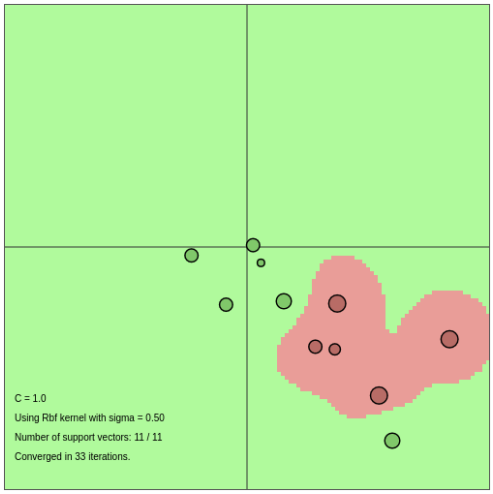
(a) Linear kernel with standard dataset.



(b) Linear kernel with new datapoint within margin.



(c) RBF kernel with standard dataset.



(d) RBF kernel with new datapoint at cluster boundary.

Figure 3: Illustrating how adding datapoints can change the importance of support vectors. *(Top-right)* Adding a new datapoint with margins for linear kernel changes the importance of other support vectors. *(Bottom-right)* Adding a new datapoint at the boundary of a cluster in RBF kernel changes the importance of support vectors

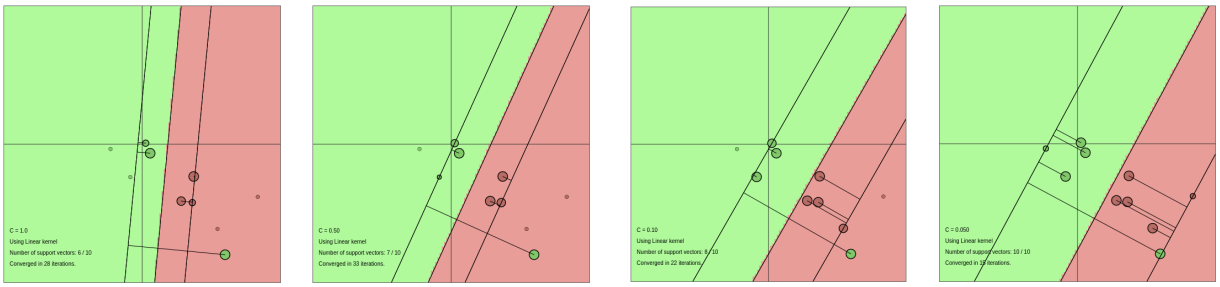


Figure 4: Linear kernel with C at values of 1, 0.5, 0.1, and 0.01.

this phenomenon [1]. They analysed the behaviours of the SVM classifier when C and/or Sigma take very small or very large values, to understand the hyperparameter space that will lead to efficient heuristic ways of searching for points in that space with small generalisation errors. They demonstrate mathematically that the RBF kernel is equivalent to a linear SVM when $\sigma \longrightarrow \infty$.

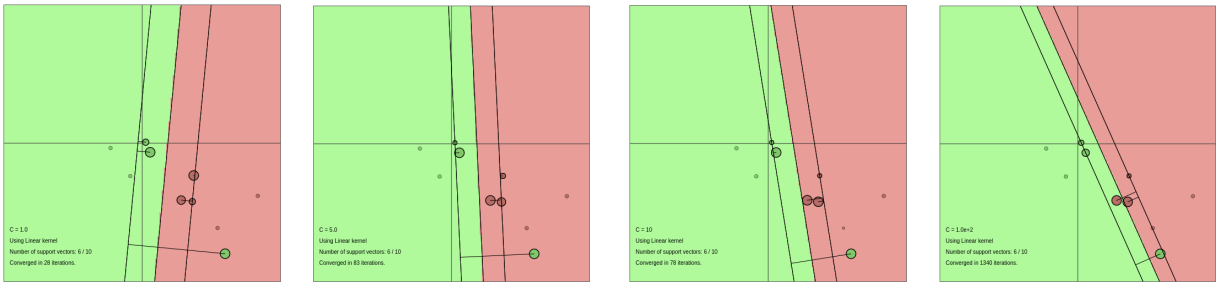


Figure 5: Linear kernel with C at values of 1, 5, 10, and 100

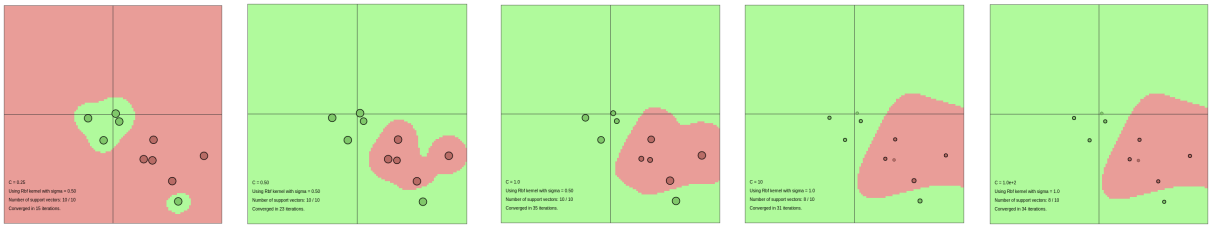


Figure 6: RBF kernel with C at values of 0.25, 0.5, 1, 10, and 100. Increasing the value of C clearly decreases the importance of each data point, and the number of support vectors.

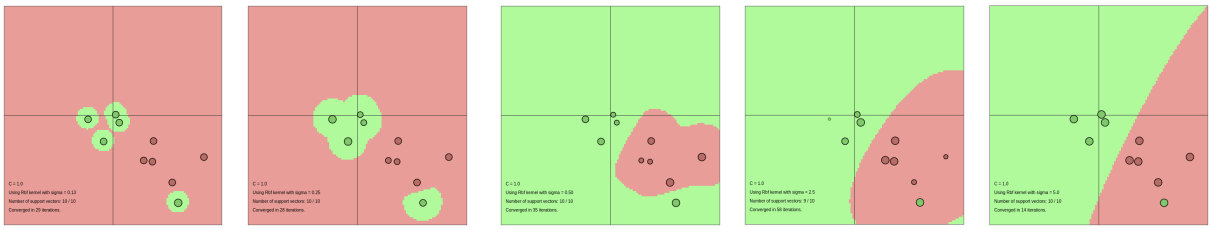


Figure 7: RBF kernel with Sigma at values of 0.13, 0.25, 1, 2.5, and 5. Increasing the value of Sigma clearly decreases the range of effect of each datapoint, ultimately resulting in a linear classification boundary at high values of sigma.

3 Least Squares SVM Classifier

Try out a polynomial kernel with different degrees. Assess the performance on the test set. What happens when you change the degree of the polynomial kernel?
A polynomial kernel was used with degrees 1 to 10. For degrees 1 and 2, the accuracy on the test set was 45% and 95%, respectively. For degrees greater than 2, the accuracy was 100%. Changing the degree of the polynomial kernel increases the flexibility of the decision boundary.

Using a RBF kernel with squared kernel bandwidth, try out a range of σ^2 and γ parameters. Assess performance on the test set and compare results with the provided script.

Figure 8 shows the accuracy of the LS-SVM over a range of possible sigma and gamma values. The results are the same as those given in the provided script.

3.1 Tuning Parameters using Validation

Compute the performance for a range of gamma and sigma values using the random split method, 10-fold CV and LOOCV. Visualise and interpret the results.

The misclassification error for the split method, 10-fold CV, and LOOCV was calculated using a grid search of sigma and gamma values between 10^{-3} and 10^{3}. Figure 9 shows the misclassification error for each grid search.

The random split naturally gives more varied accuracy than k-fold CV or LOOCV, as bias will be introduced

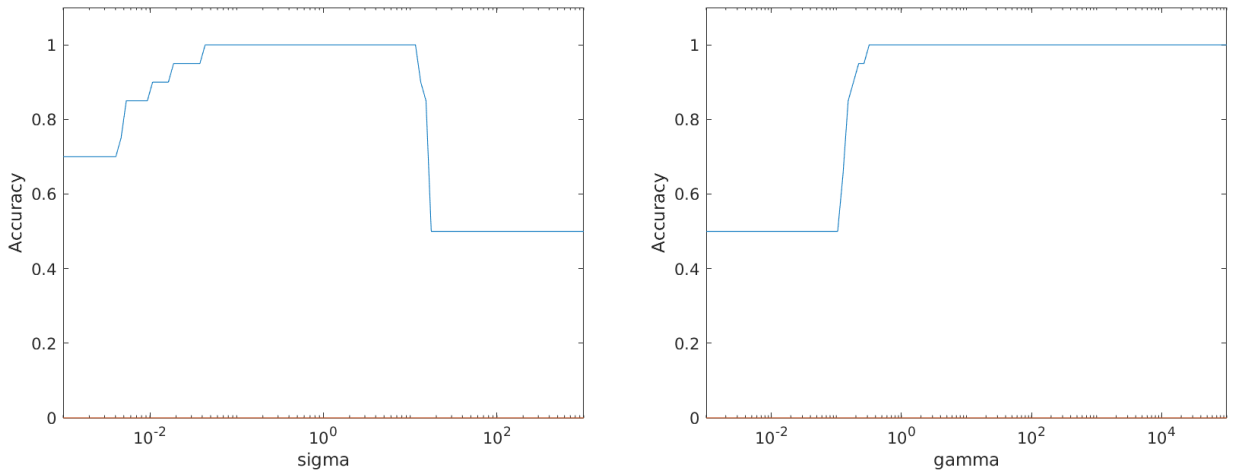


Figure 8: *(left)* Model accuracy on the test set for a range of sigma values, with gamma set to 1. *(right)* Model accuracy for a range of gamma values, with sigma set to 5.

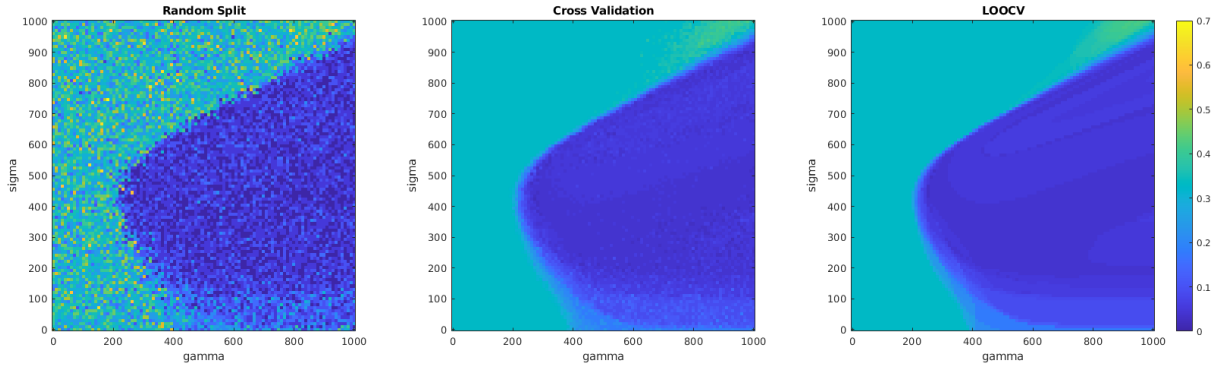


Figure 9: Colour plots showing misclassification error against gamma and sigma for different parameter tuning methods. The colour scale is shared for each plot.

based on this random selection of training and validation sets. Both k-fold CV and LOOCV show similar results, with k-fold CV showing more variability in accuracy, as evidenced by the 'spottiness' present. In LOOCV, the gradations of misclassifcation error values are more clearly delineated. This is to be expected. However LOOCV comes with increased computational cost, though that is not an issue here. There is a clear pattern in parameter space, showing decreased misclassification error for a range of sigma and gamma values.

3.2 Automatic Parameter Tuning

Try out the different 'algorithm'. What differences do you observe? Why do the obtained hyperparameters differ a lot in different runs? What about the cost? Computational speed? Explain the results.
The gamma and sigma parameters returned by both grid search and simplex vary massively between each run. The obtained parameters differ a lot in different runs because hyperparameter tuning is a non-convex optimisation problem, with many optima. The starting points given by the coupled simulated annealing are different each run, and therefore the optimum is also different due to the non-convexity of the cost-function. On average, the grid search gives slightly better performance.

Standard grid search is comparatively inefficient as it spends a lot of time investigating hyper-parameter settings that are nowhere near optimal. The efficiency of grid search can be improved by iteratively evaluating grids of increasing resolution over the parameter space (as implemented in this package). However, this is only applicable when the parameter space has low dimensionality and grid search quickly becomes computationally infeasible in higher dimensions. With NM simplex, adding an extra dimension only adds one extra point to the simplex and so it scales near-linearly with the number of dimensions. However, this optimisation problem has only two dimensions, and so this curse of dimensionality does not play a role here.

Being honest, I'm struggling to compare the computational speed of NM simplex and grid-search in low dimensions, and cannot find a definitive answer in the literature. I read Bergstra and Bengio's paper *"Random Search for Hyperparameter Optimisation"*, in which they argue that random search methods are more efficient in both high- and low-dimensional parameter spaces as not all hyperparameters are equally important to tune [2]. They state that *"grid search experiments allocate too many trials to the exploration of dimensions that do not matter and suffer from poor coverage in dimensions that are important"*, and conclude that random search methods such as NM simplex are more efficient in most cases.

As both grid search and Simplex gave similar prediction accuracies on our dataset, we can compare their computational efficiency by determining the number of cost function estimations used by each algorithm to reach this accuracy. The `tunelssvm` output for simplex showed it had 26 function estimations. The output for grid search showed that it had two iterations (each iteration has corresponds to a 'zooming' of the grid). The square root default number of function evaluations at each zoom level is given by the `grain` parameter, which has a value of 7 in the `tunelssvm` output. This means there are 49 function evaluations at each iteration. As there are two iterations for grid search on this dataset, we can conclude that the simplex algorithm took fewer cost function estimations to reach the same accuracy.

3.3 Using ROC Curves

In practice, we compute the ROC curve on the test set, rather than on the training set. Why?
We compute the ROC on the test set because it was not used to train the model in any way, and is therefore the set of data that can help us to estimate generalised performance. Computing the ROC curve on the training set would give an unrealistic estimation of how the model generalises to new data.

Generate the ROC curve for the iris.mat dataset (use tuned gam and sig2 values). Interpret the result.
Figure 10 below gives the ROC curves evaluated on the test set, using parameters estimated with both simplex (left) and grid search (right). For these specific parameters, the ROC curve for the simplex algorithm indicates perfect classification on the test data, with an AUC of 1. For the grid search, the AUC is 0.97 and has slightly worse performance.

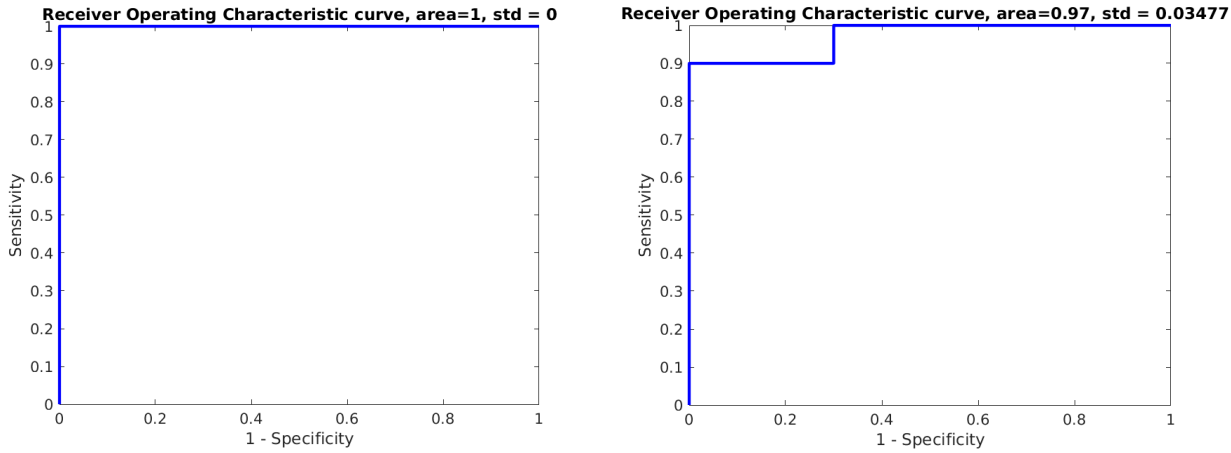


Figure 10: *(left)* ROC curves evaluated on the test set using tuned parameters from NM-simplex *(left)* and grid search *(right)*.

3.4 Bayesian Framework

The colour map indicates the probability that the instances belong a certain class. Figures 11 and 12 show the results of `bay_modoutClass` with different values of sigma and gamma, respectively. Figure 11 indicates that as sigma increases, the region of influence of each point increases, as per Equation 9. While 12 indicates that increasing gamma increases the smoothness of the surface and decreases certainty in classification away from the points.

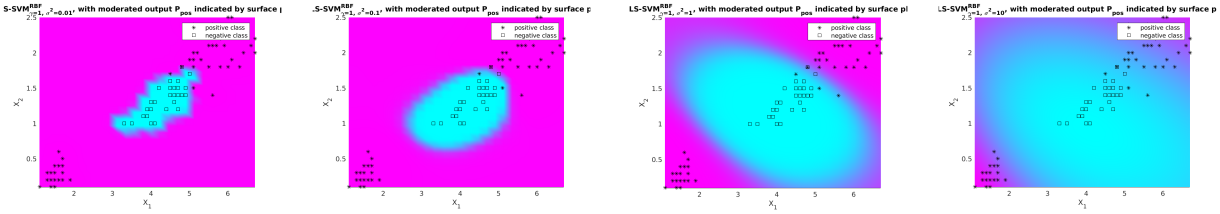


Figure 11: Colour maps indicating probability of class membership, with sigma at .001, .01, 1, and 10 (from left to right). Gamma is fixed at a value of 1.

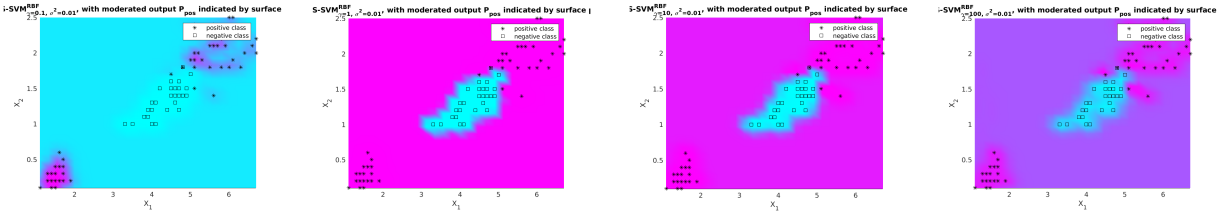


Figure 12: Colour maps indicating probability of class membership, with gamma at values of .01, 1, 10, and 100 (from left to right). Sigma is fixed at a value of 0.01.

4 Homework Problems

4.1 Ripley Data

This data can be easily visualised as it is two dimensional (see Figure 13). from this plot is is clear that x_2 is the more importance variable for classification, although x_1 is clearly still relevant. The classes are not linearly or near-linearly separable, so polynomial or RBF kernels should perform best. It is interesting that this dataset has 250 training instances and 1000 testing instances. To me, this seems like an inefficient use of data. We should instead be using at least some of this testing data to better train the model.

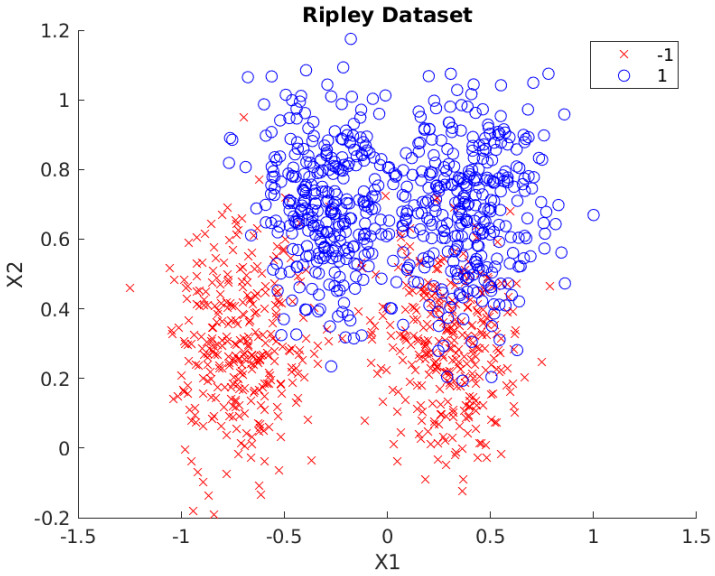


Figure 13: Scatter plot with class labels for the Ripley dataset.

The hyperparameters and kernel parameters for the linear, polynomial, and RBF models were tuned using `tunelssvm`. The resulting classifiers are visualised in Figure 14 and the ROC curves were computed for each trained model (see Figure 15). The area under the curve for each model was 0.958, 0.96, and 0.955, for linear, polynomial, and RBF, respectively. It was surprising to me that the linear model performed as well as the polynomial kernel, and better than the RBF kernel, given my initial thoughts about this data.
Are you satisfied with the performance of your model? Would you advise another methodology?

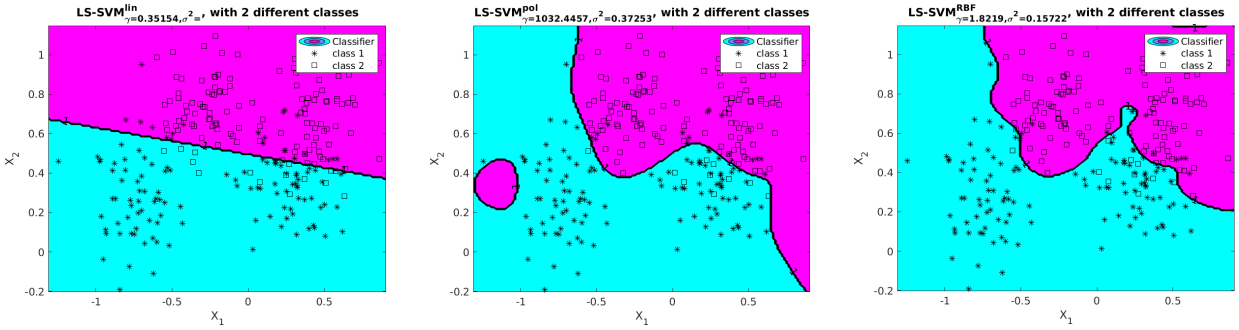


Figure 14: Classifiers for the linear, polynomial, and RBF kernels, from left to right.

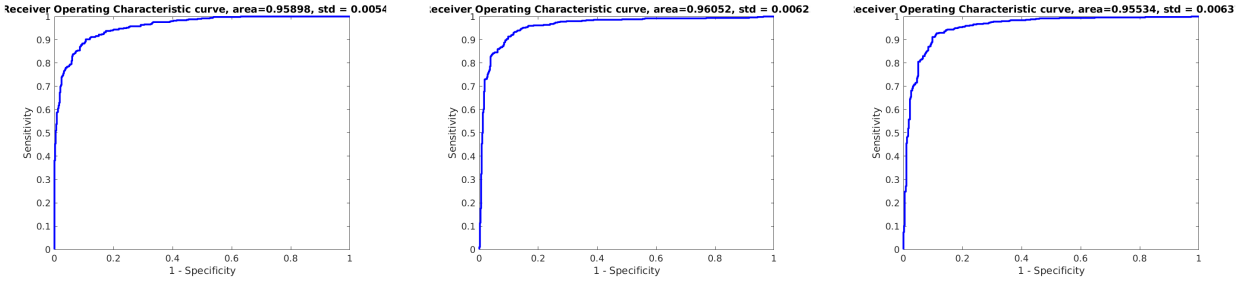
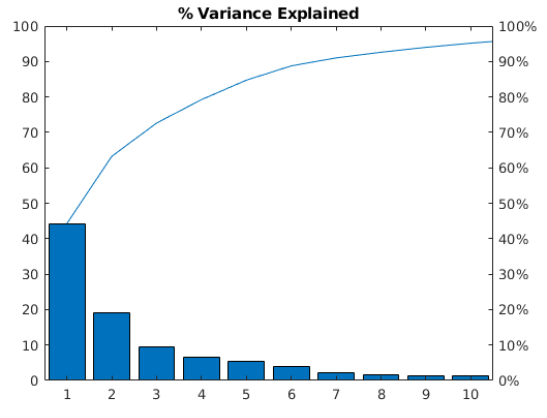


Figure 15: ROC curves for linear, polynomial, and RBF kernels, using 10-fold CV and tuned by NM-simplex.

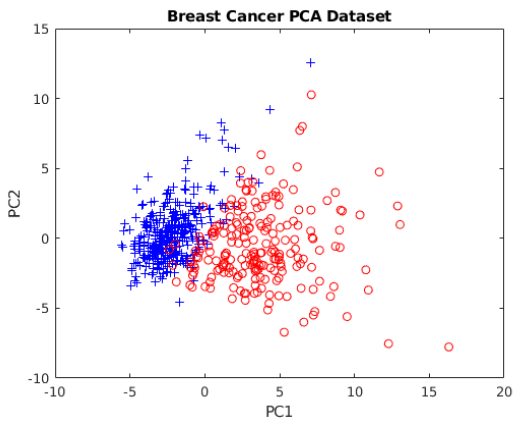
I am somewhat satisfied with the performance of the model. On the `esat.kuleuven.be` LSSVM tutorial, they achieve an AUC of 0.966, which is better than any of my models given above. I cannot think of another methodology to use, other than to change the amount of training and test data, as stated in the beginning.

5 Breast Cancer Data

The breast cancer data has 30 variables, and therefore the full dataset cannot be easily visualised. to understand the data, I first performed a PCA on the normalised data and viewed the distribution of total variance using a Pareto plot (see Figure 16a). Over 60% of the variability in the dataset is contained in the first two PCs. Plotting the transformed data in along PC1 and PC2, it is clear that the data is well-clustered and nearly linearly separable. Based on this, I think a linear model should be sufficient.



(a) Pareto plot of PCs.



(b) Scatter plot using the first two PCs.

SVMs with linear, polynomial, and RBF kernels were trained on the data using 10-fold cross validation and parameter tuning by NM-simplex. ROC curves were calculated on the test set for each model, giving AUCs of 0.994, 0.685, and 0.996, for linear, polynomial, and RBF, respectively (see Figure 17). Based on this, I would choose the linear model. It performs almost as well as the RBF kernel while being less complex and more

interpretable. When building models for medical data, interpretability and explainability is very important. We need to explain why the model gives a certain prediction. I would therefore choose the linear model in this context. The cost is also lower for the linear model, with fewer false negatives. I am satisfied with this model and would not suggest another methodology, though logistic regression could be interesting.

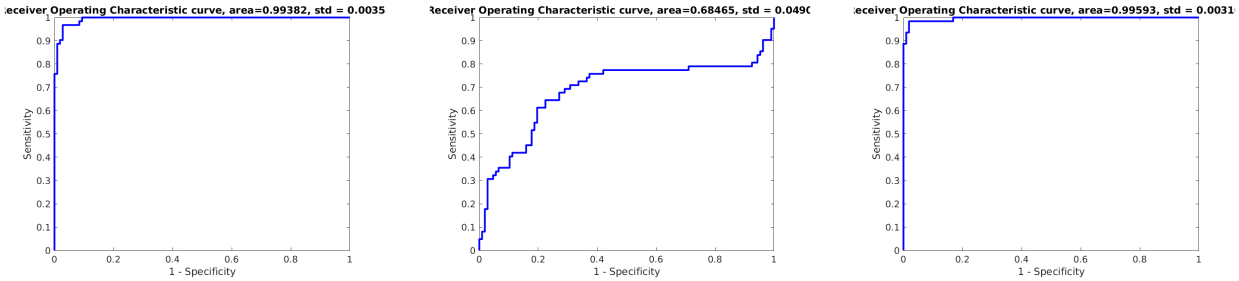
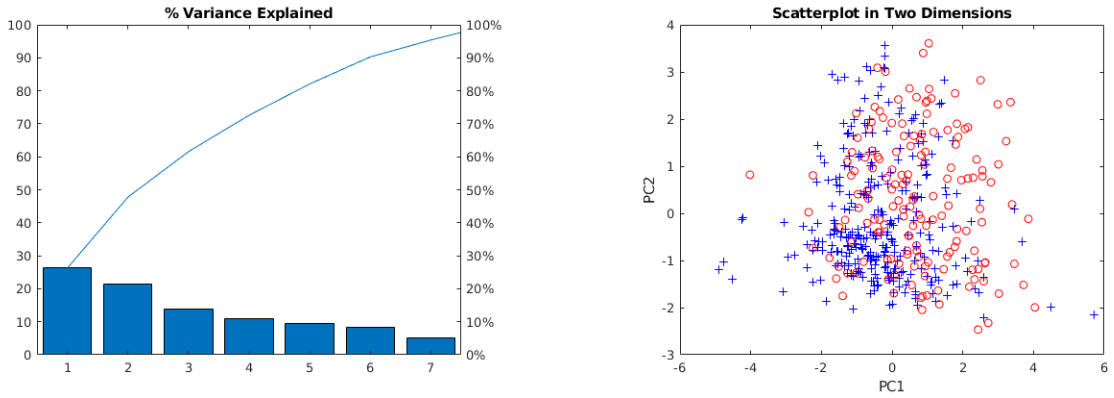


Figure 17: ROC curves for linear, polynomial, and RBF kernels (from left to right), using 10-fold cross validation parameters tuned by NM-simplex.

6 Diabetes Data

The diabetes dataset has 8 variables. Figure 18a shows that the variability present in the dataset is not concentrated on a small number of principle components, and the 95% threshold is reached only after the 7th PC. The adjacent scatter plot for the top two PCs does not show any separation between the two classes, though PC1 is clearly more informative. My intuition is that a linear model will perform poorly on this dataset, and that the RBF kernel method is most appropriate.



(a) Pareto plot of PCs.

(b) Scatter plot using the first two PCs.

Figure 19 shows the ROC curves for the three models, with AUCs of 0.84, 0.8, and 0.82 for linear, polynomial, and RBF respectively. I am surprised that the linear model outperforms the RBF kernel, given how the data appears to be structured into two overlapping clouds, with no clear linear separability between the classes.

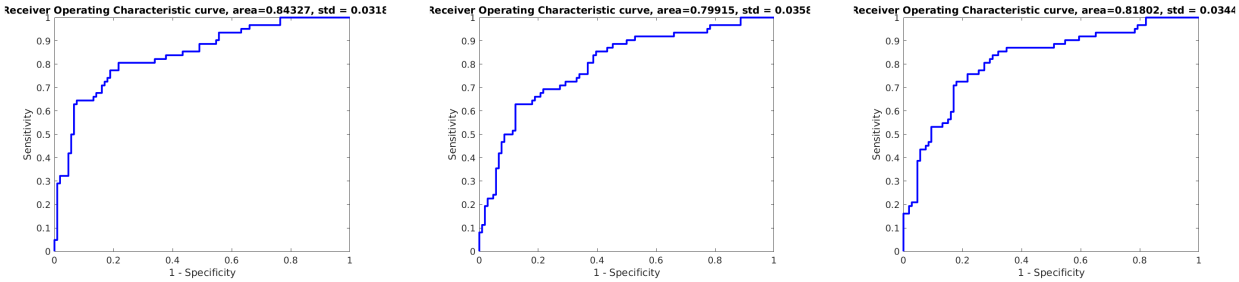


Figure 19: ROC curves for linear, polynomial, and RBF kernels (from left to right), using 10-fold cross validation parameters tuned by NM-simplex.

References

[1] S Sathiya Keerthi and Chih-Jen Lin. Asymptotic behaviors of support vector machines with gaussian kernel. *Neural computation*, 15(7):1667–1689, 2003.

[2] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.