4. Linear Regression with non-linear functions

4.1 Performing linear regression with polynomials and radial basis functions

The function $y(x) = \sin(4\pi x) + N(0, 0.1)$ on the interval $0 \leq x \leq 1$ was used to generate the the data set which the regression models would be trained on. Before training the models, the data was transformed by translating each point along the x-axis so that the data was centred about the origin. This was needed since the solutions of the regularised regression models aren't equivariant under scaling of the inputs (T.Hastie, 2009).

The regression models took the vector-matrix form $y = Xw$ where X was the design matrix and w was the weight vector containing the parameters to be learnt. Equation 1 shows the regularised sum of squares loss function which was used to model the error between the target data, t, and the regression model's prediction. The total number of data points was represented by N and λ represented the complexity parameter. The complexity parameter was used to prevent overfitting. The aim of machine learning is to provide a way of making predictions. When models use large values in the weight vector, the loss function can be optimised but the model will miss the underlying function which generates the data. The model needs to capture the underlying function if it's going to make good predictions. By adding the complexity parameter to the loss function, models which tried to overfit the data were penalised.

$$L = (t - Xw)^T(t - Xw) + \lambda\|w\|^2$$

Equation 1: Regularised sum of squares loss function

The weight vector which optimised the loss function was found by differentiating the loss function with respect to the weight vector and solving for the weight vector when $\frac{\partial L}{\partial w} = 0$.
The solution is shown in Equation 2 and applied to all regression models of the form $y = Xw$.

$$\hat{w} = (X^T X + \lambda I)^{-1} X^T t$$

Equation 2: Optimal weight vector for minimising the sum of squares

The first regression model to be evaluated was the polynomial basis function. The design matrix was of the form Nxp, where the number of rows was determined by the number of data points and the number of columns was the chosen order of the polynomial. Hence the design matrix took the form shown in Equation 3.

$$X = \begin{matrix} x_1^1 & \dots & x_1^p \\ \vdots & \ddots & \vdots \\ x_N^1 & \dots & x_N^p \end{matrix}$$

Equation 3: Design matrix of the polynomial basis function

For a training set of N=1000, Figure 1 shows the polynomial basis models from order 2 to 9 plotted against the training data when the complexity parameter was set to zero. The basis from order 5 to 9 visually appeared to replicate the underlying sinusoidal function which generated the data. The basis of order 2, 3 and 4 didn't approximate the sinusoidal well. This was because a linear combination of 1st, 2nd, 3rd and 4th order polynomials didn't provide enough freedom to model the oscillatory behaviour of the sine wave.

The second model to be evaluated was the radial basis function. The design matrix was of the form Nxp; this time, p represented the number of Gaussian functions used in the model. Each Gaussian

function had a randomly chosen mean from the transformed interval and they all had the same variance of 0.1. Equation 4 demonstrates the form of the design matrix.

$$X = \begin{matrix} \varphi(x_1, x_1) & \dots & \varphi(x_p, x_1) \\ \vdots & \ddots & \vdots \\ \varphi(x_1, x_N) & \dots & \varphi(x_p, x_N) \end{matrix} \quad \text{where } \varphi(x_i, x_n) = N(x_i, 0.1) \text{ for n } = 1, 2, \dots, \text{N}$$

Equation 4: Design matrix of the radial basis function

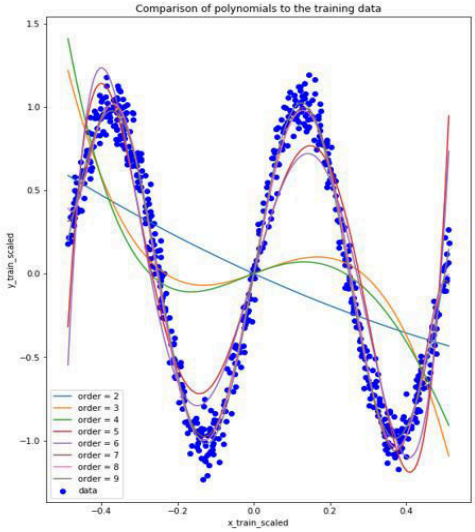

Figure 1: Polynomials of order 2 to 9 plotted against training data

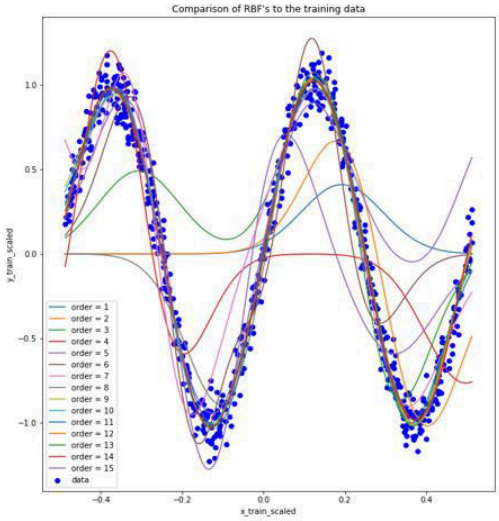

Figure 2: Radial basis functions of order 1 to 15 plotted against the training data

For a training set of N=1000, Figure 2 shows the set of radial basis functions where the order represented the number of Gaussian distributions summed as a linear combination to approximate the data and the complexity parameter was set to zero. This basis didn't appear to replicate the sinusoid well until it had an order of 9 or above. The radial basis is a linear combination of localised functions, each centred around a mean sampled from a uniform distribution. More of these functions were needed to have a global effect on the approximation of the model to the data in comparison to the global polynomial basis function.

4.2 How does regression generalise? Computing bias and variance

To quantify how the regression models generalised, the mean square error between the model and the test data was plotted against the number of parameters. This is shown for the polynomial basis and the radial basis in Figure 3 and 4 respectively.

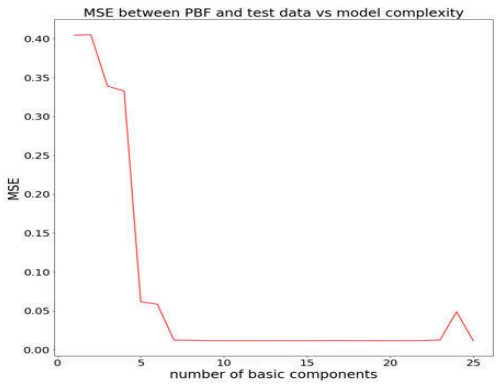

Figure 3: The MSE of the polynomial basis functions up to 25 components when λ=0

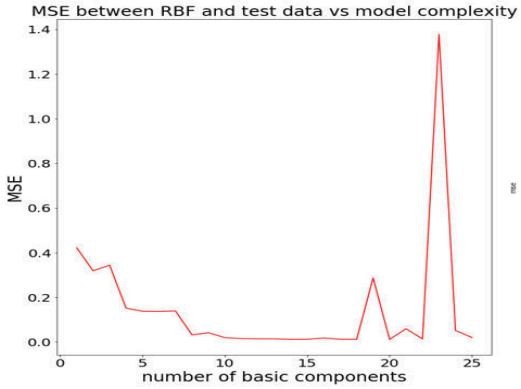

Figure 4: The MSE of the polynomial basis functions up to 25 components when λ=0

The polynomial basis depicted a steep decline in the mean square error between 2 and 5 components which supports the evidence shown in Figure 1. The increase in components was needed to give the model the capability to fit to the data. Once it had the capability, the additional parameters optimised the fit even further between 5 and 7 components after which the error reached its minimum. There was a spike in the error at 24 components which was caused by overfitting of the model. This spike was filtered out when the complexity parameter was increased as shown in Figure 5.

The radial basis showed a similar trend. There was a steep decline in the mean square error between 1 and 8 components of similar magnitude to the decline in the polynomial model between 2 and 5 components. More components were required for the radial basis to give the model the capability to fit the data because of the localised nature of the Gaussian function. The error continued to decrease until it reached its minimum error after the model had 10 components. In this model, the error spiked at 19 and 23 components; the spikes were of much larger magnitude which suggested the localised basis was more susceptible to overfitting than the global polynomial basis. Figure 6 shows the spikes were filtered out when the complexity parameter was increased.

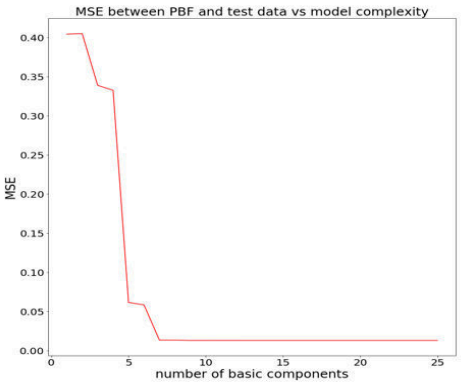

Figure 5: The MSE of the polynomial basis functions up to 25 components when $\lambda=10^{-6}$

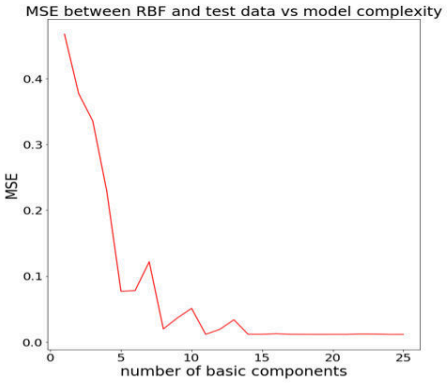

Figure 6: The MSE of the radial basis functions up to 25 components when $\lambda=10^{-5}$

The mean square error was plotted against the complexity parameter to find its optimal value. This was done for the models with the best performing number of basic components. 8 components were used for the polynomial basis and 15 components for the radial basis. The results are shown in Figures 7 and 8 respectively.

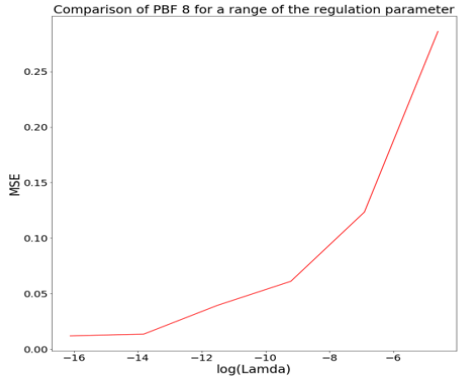

Figure 7: The MSE of the 8-component polynomial basis function for a range of λ

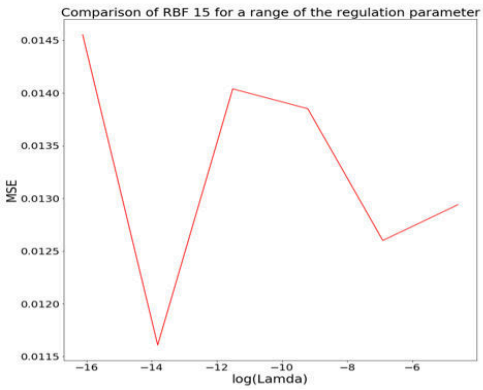

Figure 8: The MSE of the 15-component radial basis function for a range of λ

The mean square error was at its minimum for the polynomial basis when $\lambda = 10^{-8}$ which was the minimum tested value. This suggested that for a model comprised of 8 polynomial basis components, there was very little overfitting for the data set of size N=1000. For the same data set, the mean square error of the radial basis with 15 components was at its minimum when $\lambda = 10^{-7}$ although, the model was insensitive to the change in λ across the range. This suggested that the weights used in the radial basis model were generally small. Both basis functions have their advantages. The polynomial basis achieves a low error with a fewer number of components. This would be useful when computational expense needs to be reduced. The radial basis achieves a more accurate fit and is less sensitive to overfitting; however, this requires more components and will be more computationally expensive.

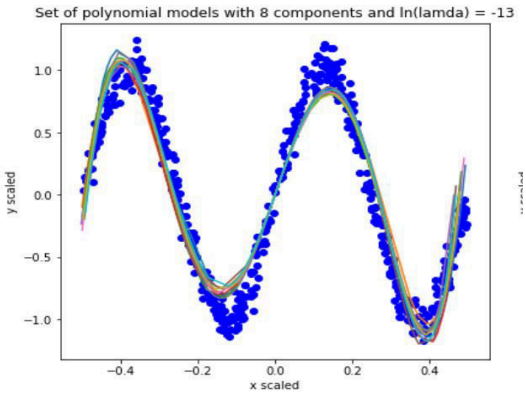

Figure 9: Set of polynomial basis models against the training data

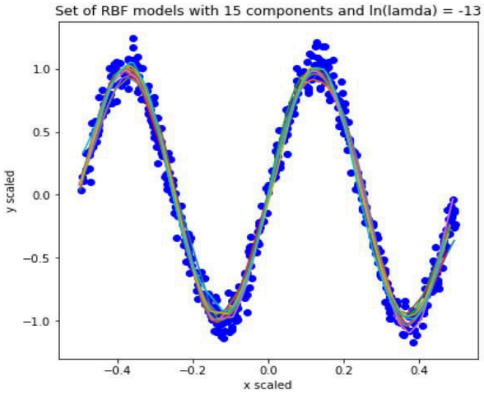

Figure 10: Set of radial basis models against the training data

Instead of using one large training set to obtain a single model, the training set could be broken down into smaller sets to obtain a set of models which an average model could be calculated from. This was investigated to see whether the single large training set or the multiple smaller training sets produced more accurate models. A training set of N=1000 was used, like with the single training set,

but this was randomly split into a set of 10 training sets, each with N=100. Figure 9 and 10 show the plots of the set of models for each basis function.

The set of radial basis models approximated the underlying function of the data better than the polynomial basis since the polynomial basis models cut off the peak and trough of the sinusoid where as the radial basis models didn't. Among several possible reasons for this, one was that the chosen complexity parameter was a better choice for the radial basis than it was for the polynomial basis. This issue was investigated by considering the bias-variance trade off.

The performance of each basis was assessed by calculating the mean model and using that to determine the squared error between the entire training set (N=1000) and the model.
The mean of the squared error was taken with respect to the data. This can be split up into two terms- the bias and the variance as shown in Equation 5. The first term is the squared bias and the second is the variance.

$$E_x[\{y(w;x) - t(x)\}^2] = \{E_x[y(w;x)] - t(x)\}^2 + E_x[\{y(w;x) - E_x[y(w;x)]\}^2]$$

Equation 5: The mean squared error in terms of the bias and the variance

The squared bias term represented the extent to which the average prediction differed from the data. The variance represented the variation between each model prediction and the mean model prediction. This was summed over all the data points to give the average mean squared error for the data set. The bias was approximated by the sample variance between the mean model and the data.

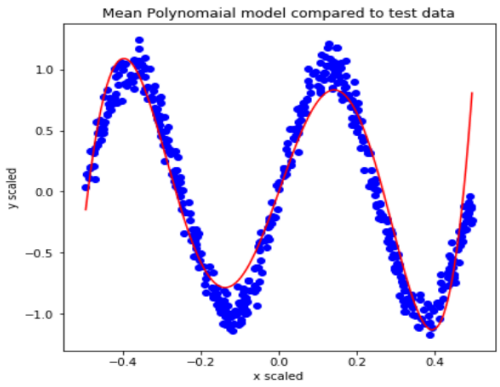

Figure 11: Mean 8 component polynomial basis with $\lambda=10^{-6}$

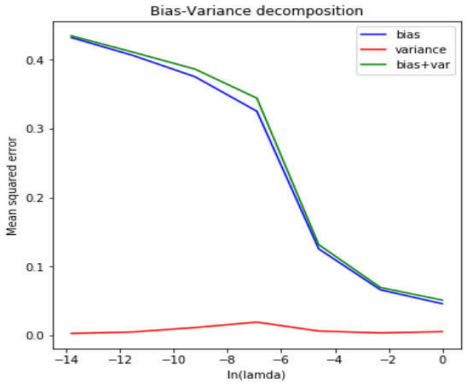

Figure 12: Bias-variance decomposition of the polynomial basis

The variance was approximated by the average sample variance of each model and the mean model. The dependence on the complexity parameter is made explicit when the loss function in Equation 1 was substituted into the bias and variance terms. In the case of the bias term, Xw represented the prediction of the mean model. In the case of the variance term, t represented the mean model's prediction and Xw represented the individual model's prediction. Figure 11 shows the plot of the mean polynomial basis and Figure 12 shows the polynomial basis' expected loss split into its bias and variance term against the complexity parameter.

The mean model was as expected since the set of polynomials cut off the peaks and troughs of the sinusoid. Figure 9 shows the set of polynomials were tightly plotted together when ln(λ)=-13, this was supported by the bias variance decomposition. As the complexity parameter increased, the bias decreased, and the variance remained minimal. This suggests that a large complexity parameter would lead to better performance when training a set of models of smaller training sets. This was unexpected since the results above showed a small variation in the set of models and the average

model had a poor fit. It was expected that the better fits occurred when the set of models had a larger variation.

5. Classification

5.1 Data 1: Separate 2 Gaussians

The purpose of this section was to find a decision boundary between the two classes so new data points could be classified correctly. Two approaches were trialled. The first involved finding a discriminant function which mapped each input to one of the classes. The second was a probabilistic approach where the posterior distributions of each class were modelled, and a new data point would be assigned to the class with the higher probability.

The generated data sets used for this part of the report are shown in Figure 13 and Figure 14. 180 data points were independently sampled from two different gaussian distributions; 100 points from the distribution corresponding to class A and 80 points from the distribution corresponding to class B.

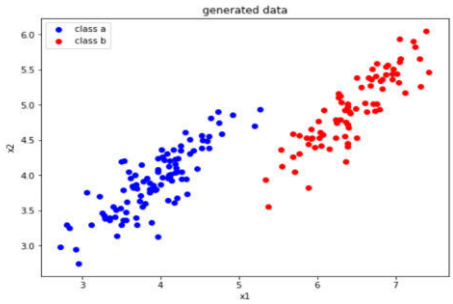

Figure 13: Generated data set with covariance matrices $S_A = S_B = \begin{matrix} 0.25 & 0.21 \\ 0.21 & 0.25 \end{matrix}$

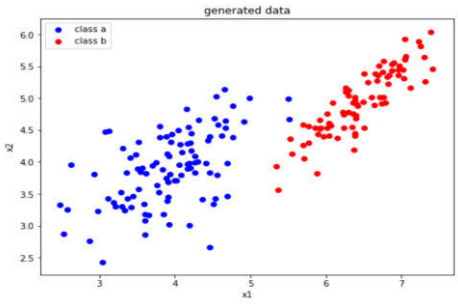

Figure 14: Generated data set with covariance matrices $S_A = \begin{matrix} 0.34 & 0.2 \\ 0.2 & 0.4 \end{matrix} and\ S_B = \begin{matrix} 0.25 & 0.21 \\ 0.21 & 0.25 \end{matrix}$

The model $y = w^T x$ was used to determine the discriminant function. The value of y represented the projection of the data from 2-d space (x1-x2 plane) on to a 1-d space (w-axis). No bias term was included in the model; hence the decision boundary was made up of the locus of points which satisfied the condition $y = 0$. Figure 15 shows histograms for the projection of the input data from Figure 13 on to two different axes. The results showed there was a separation between the two classes when $w = [-2, 2]$, whereas the projection of the data on to $w = [0, 1]$ lead to significant overlap. These results were expected since the weight vector, w, must be perpendicular to the decision boundary (C.Bishop, 2006) and by looking at Figure 13, it was clear to see the approximate position the boundary needed to take.

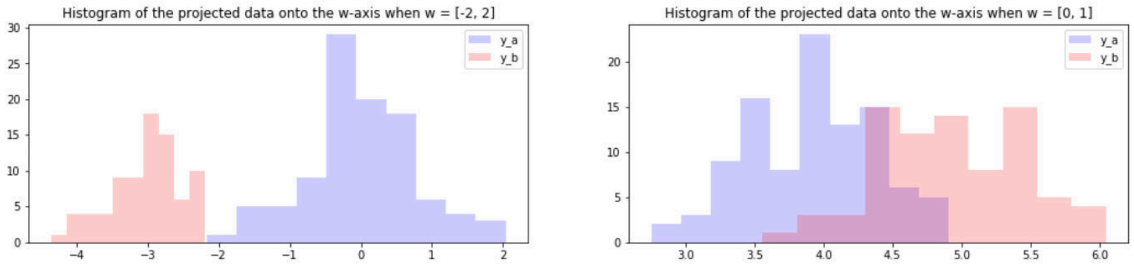

Figure 15: Histograms showing the projected component of each data point onto the w-axis vs the number of data points with that projected component

The Fisher criterion was used to determine the axis which maximised the separation between the two classes. The Fisher criterion returned a scalar which represented the ratio of the between class variance (separation of means) and the total within class variance (sum of the class variations). By modelling it as a function of the weight vector, the axis which maximised the ratio was found. A change of basis was used to find the axis which maximised the ratio. The new basis used for the weight vector was an orthonormal set dependent on the angle between the weight vector and the x1-axis. This transformation meant some arbitrarily chosen starting vector, w(0), could be rotated through 2π radians without being scaled and return a value of the Fisher criterion for each direction. Figure 16 shows the resultant plot of the Fisher criterion as a function of this angle. The criterion had two directions which maximised the output, this seemed reasonable as there are two perpendicular directions for all possible axis in a plane. The parameterised weight vector, w(θ), which corresponded to one of the directions that maximised the Fisher criterion was $w = [0.83, -0.56]$. The histogram of the data projected on to this axis is shown in Figure 17.

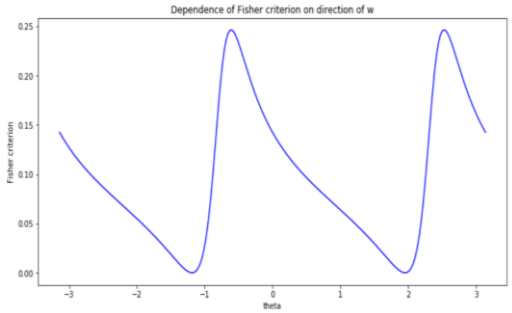

Figure 16: Fisher criterion rotated between -π and π

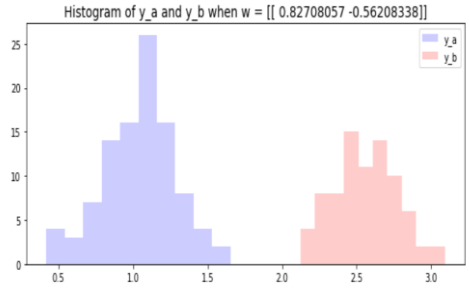

Figure 17: Histogram of data projected on to the axis of maximum separation

To classify the data using this model, a bias term must be added so that the model becomes $y = w^T x + w_0$. The parameter w_0 should take the value at the centre of the gap between the two classes. The result of the projection on to this axis would be used to classify the data as follows: $if\ y > w_0\ , x \in x_B$ and $if\ y < w_0,\ x \in x_A$. By taking the value at the centre of the gap, the misclassification rate would be minimised. Using a larger sample size would make this method more transferable to a test data set. As the training data sets were known to be normally distributed, the projection on to the w-axis should be normally distributed. The histograms shown in Figure 17 suggest that more data points were needed to get a better estimation of the distribution of the data along the w-axis.

In this example, the data sets were generated so the likelihood probability distributions, $P(x|C_A) and\ P(x|C_B)$, were known. These were the Gaussian probability distributions the data was sampled from. Figure 18 shows the contour plot of the two probability distributions when the covariance matrices were equal. The contours on the left are from class A and the contours on the right are from class B. The arrow indicates the optimal direction of the weight vector calculated earlier.

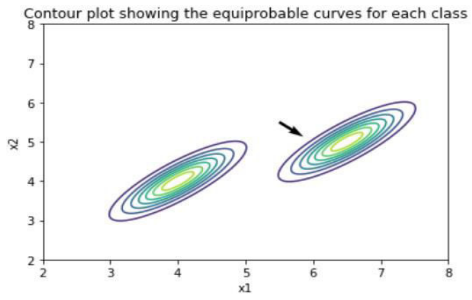

Figure 18: Contour plot showing the equiprobable curves for class A and class B. The arrow indicates the direction of optimal w

The log-odds method was used to find the decision boundary when the probabilistic approach was used. This involved taking the natural log of the ratio of class posterior distributions. Bayes' theorem was used to calculate the posterior probability distributions, $P(C_A|x)\ and\ P(C_B|x)$. The prior

distributions, $P(C_A)\ and\ P(C_B)$, were calculated from the ratio of the number of data points in the class to the total number of data points sampled. When the posterior probabilities were equal, the log-odds returned the value zero (S.Rogers, 2016). In practice, a tolerance of ± 0.1 was accepted to account for the floating-point accuracy. The decision boundary was the set of points which mapped to zero under the log-odds transformation. Figure 19 and Figure 20 show the decision boundary for the posterior distributions when the covariance matrices were equal and different respectively. All other points in the input space were classified to the class with higher probability. This was inferred from the sign of the log-odds.

The decision boundaries demonstrated were as expected. When the log-odds of Gaussian probability distributions were computed with equal covariance matrices, the quadratic term in x (from the quadratic argument of the exponential term) cancelled out, resulting in a linear boundary. However, when the covariance matrices differed, the quadratic term remained, hence the quadratic shape of the boundary.

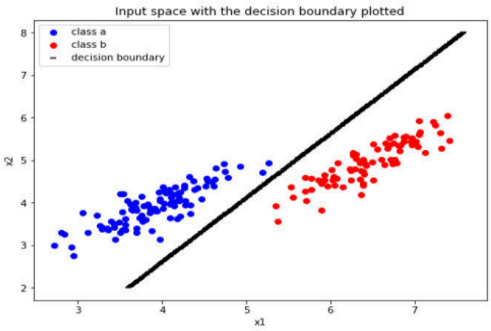

Figure 19: Decision boundary for covariance matrices

$$S_A = S_B = \begin{matrix} 0.25 & 0.21 \\ 0.21 & 0.25 \end{matrix}$$

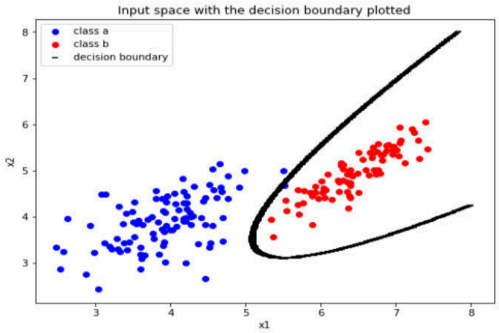

Figure 20: Decision boundary for covariance matrices

$$S_A = \begin{matrix} 0.34 & 0.2 \\ 0.2 & 0.4 \end{matrix} and\ S_B = \begin{matrix} 0.25 & 0.21 \\ 0.21 & 0.25 \end{matrix}$$

5.2 Data 2: Iris data

The Iris dataset was made up of 150 data points equally split between 3 classes. Each data point was a 4-dimensional vector. This section experimented with the use of Linear Discriminant Analysis (LDA) for projecting the 4-dimensional input vector on to a lower dimensional space with better separation of the three classes. The lower dimensional space provided a better perspective for classifying the data.

To begin this analysis, the Fisher ratio was used as the function to optimise the separation of the classes. The Fisher ratio as a function of the weight vector was modelled by Equation 6. As mentioned earlier, it is a ratio of the separation of means to the spread of the classes. In the generalised case, where there were more than two classes, a mean of the class mean vectors was used as the centre point for determining the between class variance. The within class variances were determined in the usual manner.

$$F(w) = \frac{w^T S_B w}{w^T S_w w} \text{ where } S_B = \sum_k (m_k - m)(m_k - m)^T \text{ and } S_W = \sum_{n \in C_k} (x_n - m_k)(x_n - m_k)^T$$

Equation 6: Fisher criterion as a function of w. S_B represents the between class covariance matrix where m_k represents the class mean and m represents the mean of class means. S_W represents the sum of the within class covariance matrices and k is the class number, $k = 0, 1, 2$

The purpose of this part of the analysis was to find a basis of weight vectors which had the best combination of maximising the separation of means and minimising the sum of class variances. These weight vectors were stacked into a weight matrix of the form kx4, where k<4 (hence the weight matrix transforms the input data from a 4-dimensional input space to a k-dimensional space with optimised separation). To find this set of optimal weight vectors, the derivative of F was calculated using the quotient rule as shown by Equation 7.

$$\frac{\partial F}{\partial w} = \frac{2}{(w^T S_W w)^2}\{(w^T S_W w) S_B w - (w^T S_B w) S_W w\}$$

Equation 7: Derivative of the Fisher ratio with respect to the weight vector

To find the maximised Fisher ratio, Equation 7 was solved for the case when $\frac{\partial F}{\partial w} = 0$. This led to Equation 8.

$$(w^T S_W w) S_B w = (w^T S_B w) S_W w$$

Equation 8: Condition when the derivative of the Fisher ratio is maximised

The quadratic terms in the brackets were scalers, so both sides were divided by the quadratic term on the left, the transformation $S_W^{-1}$was then applied to the left of both sides of the equation which led to the generalised eigenvalue problem shown in Equation 9.

$$(S_W^{-1} S_B) w = \lambda w \text{ where } \lambda = \frac{w^T S_B w}{w^T S_W w}$$

Equation 9: The generalised eigenvector equation arising from maximising the Fisher ratio

The eigenvalue problem was then solved in the usual way by finding the eigenvalues and eigenvectors of the matrix $S_W^{-1}S_B$. The resulting eigenvalue and eigenvector pairs are shown in Figure 21.

```
Computed Eigenvalues of the generalised eigenvalue problem:
 [6.43838584e-01 5.70782085e-03 1.03566112e-16 2.93263314e-17]

Rows of the matrix are the Eigenvectors corresponding to the Eigenvalue of the same index:
 [[ 0.20874182  0.38620369 -0.55401172 -0.7073504 ]
 [-0.00653196 -0.58661055  0.25256154 -0.76945309]
 [ 0.36980581  0.18417176  0.2575547  -0.87349299]
 [-0.48455506  0.44109192  0.48751751 -0.57703639]]
```

Figure 21: The Eigenvalue and Eigenvector pairs from the generalised Eigenvalue problem

Since S_W and S_B described the data set, the eigenvectors of $S_W^{-1}S_B$ were considered as the representation of the data sets principal axes and the eigenvalues represented the magnitude of each axis. Two of the axes had magnitudes which tended to zero; hence they held little information about the data set. Since the other two axes had a larger magnitude, they contained a lot of the information about the data. Because of this, the eigenvectors corresponding to the two largest eigenvalues were used to form a basis for transforming the data from 4 dimensions to 2. This transformation was encoded by forming a 2x4 matrix, where row 1 corresponded to the first eigenvector in Figure 21 and row 2 corresponded to the second. The projected data on to these axes is shown in Figure 22.

Figure 22 displays a good separation of classes along the y1 axis. The y2 axis doesn't separate the classes, this seems reasonable since the first eigenvalue is two orders of magnitude larger than the second. For this reason, the data was projected on to the eigenvector corresponding to the largest eigenvalue as shown in Figure 23.

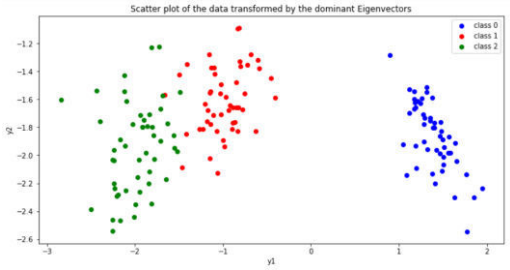

Figure 22: Scatter plot of the data projected on to 2-dimensional space

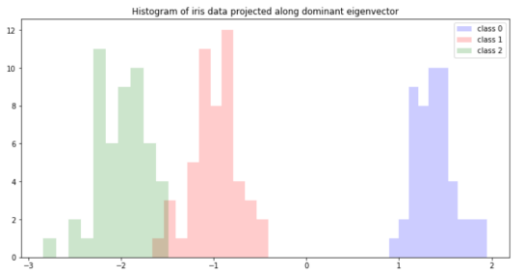

Figure 23: Histogram of the 4-d iris data projected on to the dominant eigenvector

Two methods have been shown for computing the optimal weight vector which maximised the Fisher ratio. In the last section, the weight vector was rotated in the plane and the Fisher ratio was computed for all possible vectors. This worked well for a 2-dimensional case, but since it involved directly calculating the Fisher ratio of every weight vector, it would not be feasible to use this method in high dimensional spaces. Finding solutions to a generalised eigenvalue problem is a more suitable method for higher dimensional spaces as it allows the classification problem to be solved in a lower dimension which is less computationally expensive.

References

C.Bishop, 2006. In: *Pattern Recognition and Machine Learning.* s.l.:Springer, pp. 181-182.

S.Rogers, M., 2016. In: *A First Course in Machine Learning.* s.l.:CRC press, pp. 178-179.

T.Hastie, R. J., 2009. In: *The Elements of Statistical Learning.* New York: Springer, pp. 61-62.