COMP3223

Coursework

Alberto Tamajo

Student ID: 30696844

December 2021

Electronics and Computer Science Department
University of Southampton

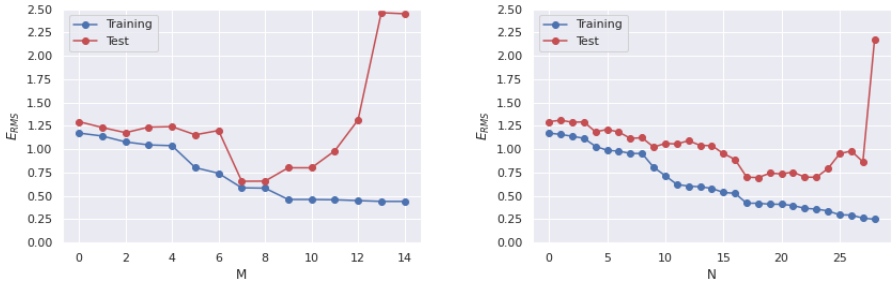


Figure 1: The left and right graphs illustrate the dependence of the generalisation performance on the polynomial order M and the number of Gaussian Radial basis functions N, respectively. The left plot shows that values of M in the range $0 \leq M \leq 4$ perform rather poorly both in the training and test dataset as these polynomials are incapable of capturing the oscillation of the underlying function. The training loss sharply decreases when $5 \leq M \leq 8$ and then decreases more steadily. On the other hand, the test loss reaches its minimum at $M = 8$ and then shows a sharp surge. This phenomenon is called overfitting, and it is caused by the fact that as M gets larger, the resulting polynomial increasingly captures the noise on the training data. As a consequence, the performance on the training data becomes increasingly better, but the generalisation performance constantly decreases. Based on this graph, $M = 8$ gives the best generalisation performance. The right plot shows that Gaussian RBFs are also affected by overfitting when $N > 18$. The N=18-radial achieves the best generalisation performance but the performance of the M=8-polynomial is superior.

1 Linear regression with non linear functions

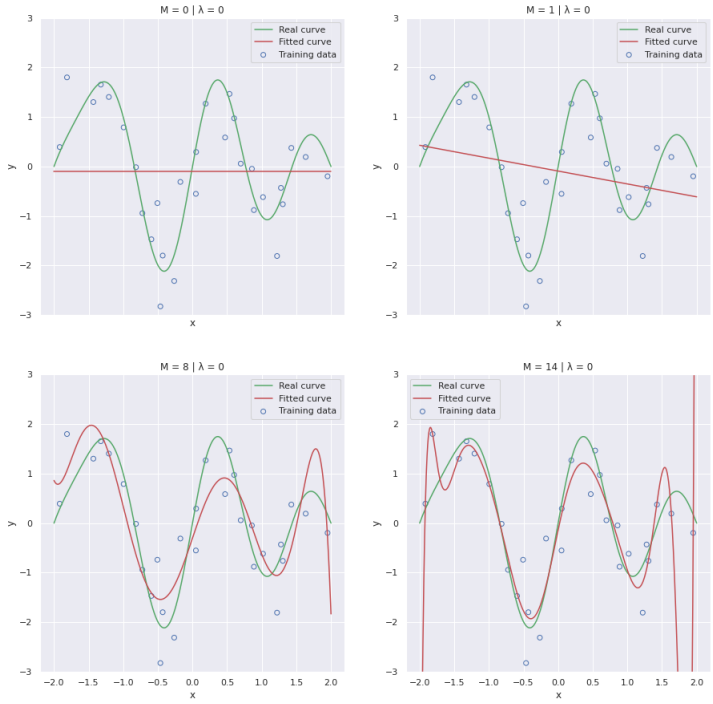
For this exercise, the training and test datasets have been generated following the instructions of the exercise description and using a Gaussian noise with mean 0 and variance 0.5 to create very noisy data that makes the task of achieving generalisation a challenging one. This report uses the notation "N-radial" to indicate a function consisting of N Gaussian Radial basis functions. The basis function centres are selected randomly from the x-ordinates of the training data guaranteeing that, given an N-radial and an L-radial, the first m^{th} centres of both functions are equal.

In the case of polynomial and Gaussian radial fitting, the goal of model selection revolves around finding which order M and which number of Gaussian radial basis N gives the best generalisation performance, respectively. In the case of this exercise, this is achieved by checking the performance of several polynomials and several Gaussian radial basis functions both on the train and test dataset. Figure 1 shows that the $M = 8$-polynomial gives the best generalisation performance for polynomial fitting, while the $M > 8$-polynomials perform rather poorly at predicting the values for new data observations. Similarly, the $N = 18$-radial performs the best at predicting new observations for Gaussian RBFs fitting (its performance is inferior to the $M = 8$-polynomial), while the $N > 18$-radials give unsatisfactory predictions. At first glance, such a result may seem strange as high-order polynomials and high-N-radials contain low-order polynomials and low-N-radials, respectively. However, there is a straightforward reason behind such a phenomenon: **complex models are more flexible than simple ones and, as such, are more likely to capture the noise present on the training dataset**. This phenomenon is called overfitting. Figure 2 and Figure 3 give us a demonstration, in the context of this exercise's dataset, that high-order polynomials and high-N-radials tend to capture a lot of noise.

Overfitting is a problem that can be solved by adopting one of the following strategies: limiting the number of basis functions, increasing the size of the training data and regularisation.

Figure 4 illustrates that, even though the $M = 8$-polynomial achieves the best generalisation performance in the training dataset, it cannot capture the interesting properties of the real curve when new data is added. At the same time, the $M = 14$-polynomial perfectly fits the real curve. Similarly, the $N = 18$-radial does not achieve the same level of fitting to the real curve as the $N = 28$-radial when the training dataset size is increased. This shows that increasing the size of the training data really helps at reducing overfitting for the complex models and that we need to be careful at limiting the number of basis functions as this has the side effect of reducing the capability of capturing interesting properties in the data. Given that the $M = 14$-polynomial and the $N = 28$-radial seem to be complex enough to capture the properties of the real curve, it seems reasonable to try to find the optimal regularisation parameter for them and check whether they achieve a better generalisation performance in the test set than the $M = 8$-polynomial. Figure 5 and Figure 6 show that this does not actually happen. However, it is worth noticing that the generalisation performance achieved by both is significantly better than the previous performance without regularisation. Besides, the generalisation performance reached by the $N = 28$-radial is substantially better than the performance achieved by the $M = 14$-polynomial. The same process has been carried out for the $8 < M < 14$-polynomials and the $18 < N < 28$-radials, and all of of them have failed at reaching the best generalisation performance. In Figure 5 and 6, the optimal regularisation parameter is found by looking at the parameter that minimises the test loss. Equivalently, from a frequentist point of view, the optimal regularisation parameter is the one that leads the model to have the best balance between bias and variance. This is illustrated in Figure 7.

It seems reasonable to conclude that, even though all the $8 < M \leq 14$-polynomials and $18 < N \leq 28$-radials are increasingly more capable of capturing the interesting properties of the underlying curve, the $M = 8$-polynomial achieves the best generalisation performance if we limit ourselves at checking the performance reached on the generated test set. However, by the nature of this exercise, the training dataset can be increased without restraints. Figure 4 has shown that that the $M = 14$-polynomial gives a perfect fit if additional data is generated.



	M=0|λ=0	M=1|λ=0	M=8|λ=0	M=14|λ=0
w0	-0.104682	-0.0915234	-0.303005	-0.138867
w1		-0.260312	4.14089	6.606789
w2			-0.0716018	-2.791821
w3			-7.70052	-20.604497
w4			0.0441262	10.859870
w5			3.5613	26.588995
w6			0.251537	-15.870708
w7			-0.484272	-21.864297
w8			-0.0652371	11.044562
w9				10.922039
w10				-3.698234
w11				-2.800325
w12				0.547738
w13				0.275448
w14				-0.026075

Figure 2: (Left)Plots of polynomials of several orders M, shown as red curves, fitted to the training dataset (blue circles) without regularisation. The real curve is shown in green. (Right)Table of the coefficients w for polynomials of various order. The plots on the left show that the constant and first-order polynomials are a poor fit for both the training data and the real curve. The $M = 8$-polynomial seems to achieve the best balance between fitting the training data and generalising the real curve. On the other hand, the $M = 14$-polynomial better fits the training data but does not generalise well because of its excessively wiggly shape. The large oscillations of the $M = 14$-polynomial can be explained by looking at the table of coefficients. This table shows that as M increases, the magnitude of the coefficients gets larger. Therefore, the coefficients of the $M = 14$-polynomial have been tuned so that to match the training data as much as possible, but this causes large oscillations between the training points.

2 Fisher Linear Discriminative Analysis

For simplicity and without loss of generality, the Fisher linear discriminative analysis for binary classification is explained in what follows.
Let M be a D-dimensional dataset whose entries belong to either of 2 classes, C_1 and C_2. Rather than directly dividing the input space into two decision regions, the Fisher LDA first projects M down to a one-dimensional space using a vector $\boldsymbol{w}$. In other words, every entry $\boldsymbol{x}$ in M is projected onto a point $y = \boldsymbol{w}^T\boldsymbol{x}$ lying on a line parallel to $\boldsymbol{w}$. The newly produced one-dimensional dataset M' is then utilised to learn a threshold t so that all projected points $y >= t$ belong to C_1, while the projected points $y < t$ belong to C_2. Therefore, the Fisher LDA is a two-stage model combining a projection technique and a linear classification model.
The choice of the vector $\boldsymbol{w}$ significantly affects the classification performance. Indeed, if $\boldsymbol{w}$ is not carefully chosen, projecting M into a one-dimensional space may lead to a considerable loss of information, whereby classes C_1 and C_2 overlap even though they are linearly separable in the original dataset M. The idea proposed by Fisher to overcome such a problem is elegant. The best vector $\boldsymbol{w}^*$ is the vector that maximises the between-class variance and contemporaneously minimises the within-class variance. Equivalently, the vector $\boldsymbol{w}^*$ is obtained by maximising the Fisher ratio.

3 Data 1: separate 2 Gaussians

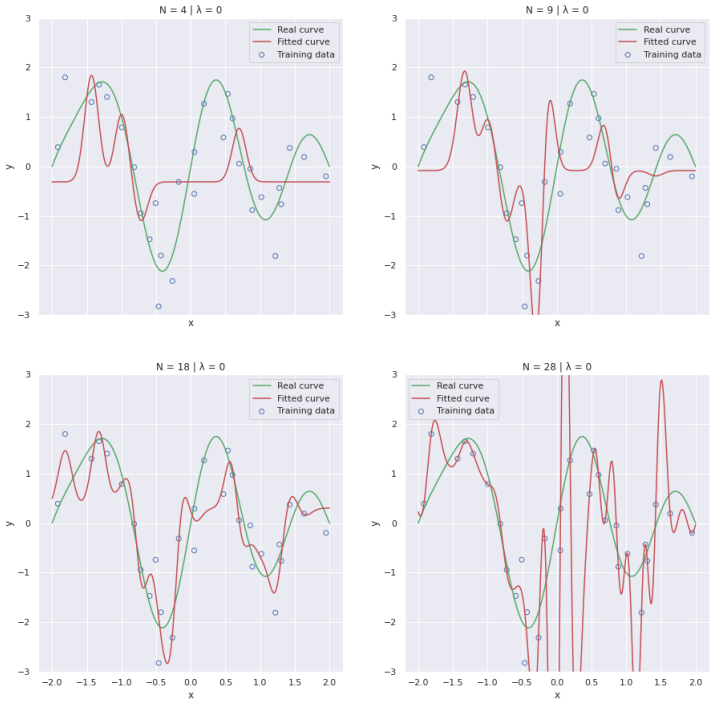
Two datasets, D_1 and D_2, have been generated by drawing samples from the following two 2-dimensional Gaussian distributions, respectively

$$\boldsymbol{x}_1 \sim \mathcal{N}(\boldsymbol{x}|\boldsymbol{m}_1, \boldsymbol{S}_1), \quad \boldsymbol{x}_2 \sim \mathcal{N}(\boldsymbol{x}|\boldsymbol{m}_2, \boldsymbol{S}_2)$$

where:

$$\boldsymbol{m}_1 = (10, 4), \quad \boldsymbol{m}_2 = (33, 15), \quad \boldsymbol{S}_1 = \boldsymbol{S}_2 = \begin{pmatrix} 10 & -10 \\ -10 & 30 \end{pmatrix}$$

Thus, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ have the same covariance matrix but different means as they are shifted by the vector



	N=1|λ=0	N=18|λ=0	N=28|λ=0
w0	-0.175371	0.302754	1.156371
w1	0.874802	-1.21542	-4.770394
w2		-2.17363	-1.598917
w3		0.31776	-0.348614
w4		0.0681269	-0.188939
w5		1.50823	0.599384
w6		-0.270847	23.987410
w7		1.43771	8.234522
w8		0.38238	9.129290
w9		-3.02842	-8.080642
w10		1.16034	1.631976
w11		1.73323	1.691338
w12		-0.321773	-669.825602

Figure 3: (Left)Plots of several N-radials, shown as red curves, fitted to the training dataset (blue circles) without regularisation. The real curve is shown in green. (Right)Table of the coefficients w for several N-radials. The table is cut at w_{12} as otherwise, it would fill too much space. The plots on the left show that the constant and $N = 1$-radial are a poor fit for both the training data and the real curve. The $N = 18$-radial seems to achieve the best balance between fitting the training data and generalising the real curve. On the other hand, the $N = 28$-radial passes through 28 training data points but does not generalise well because of its excessively wiggly shape. The large oscillations of the $N = 28$-radial can be explained by looking at the table of coefficients. This table shows that as N increases, the magnitude of the coefficients gets larger. Therefore, the coefficients of the $N = 28$-polynomial have been tuned so that to match the training data as much as possible, but this causes large oscillations between the training points.

$$(33, 15) - (10, 4) = (23, 11)$$

The covariance matrix has been chosen so that to obtain a negative correlation between the x and y coordinates. The number of generated points for both datasets is 100.

Figure 8 shows that D_1 and D_2 are two linearly separable datasets.

3.1 Exercise 1

The previous section claimed that the choice of the vector $\boldsymbol{w}$ significantly affects the classification performance of the Fisher LDA. This exercise gives a concrete demonstration of that claim's validity by selecting two vectors, $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$, that lead to two very different classification performances. While the projections of D_1 and D_2 onto $\boldsymbol{w}_1$ are still linearly separable, the same does not hold when the two datasets are projected onto $\boldsymbol{w}_2$. From Figure 8, it is noticeable that the worst possible projection for D_1 and D_2 is onto a line parallel to their stretching direction. On the other hand, the best projection is onto a line that is perpendicular to their stretching direction. The stretching direction of the two datasets has been computed by performing eigendecomposition in the covariance matrix S_1 and normalising the eigenvector $\boldsymbol{v}$ associated with the largest eigenvalue. Thus,

$$\boldsymbol{x}_2 = \frac{\boldsymbol{v}}{||\boldsymbol{v}||}, \quad \boldsymbol{x}_1 \perp \boldsymbol{x}_2 \wedge ||\boldsymbol{x}_1|| = 1$$

The vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are normalised because only their directions is important when computing the projections. The result of projecting D_1 and D_2 onto $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ are illustrated in Figure 9

3.2 Exercise 2

In the previous exercise, the projections of D_1 and D_2 onto the vector w_1 were linearly separable, while they were not when projected onto w_2. In this exercise, the values of S_a and S_b have been altered to show that different linearly separable datasets

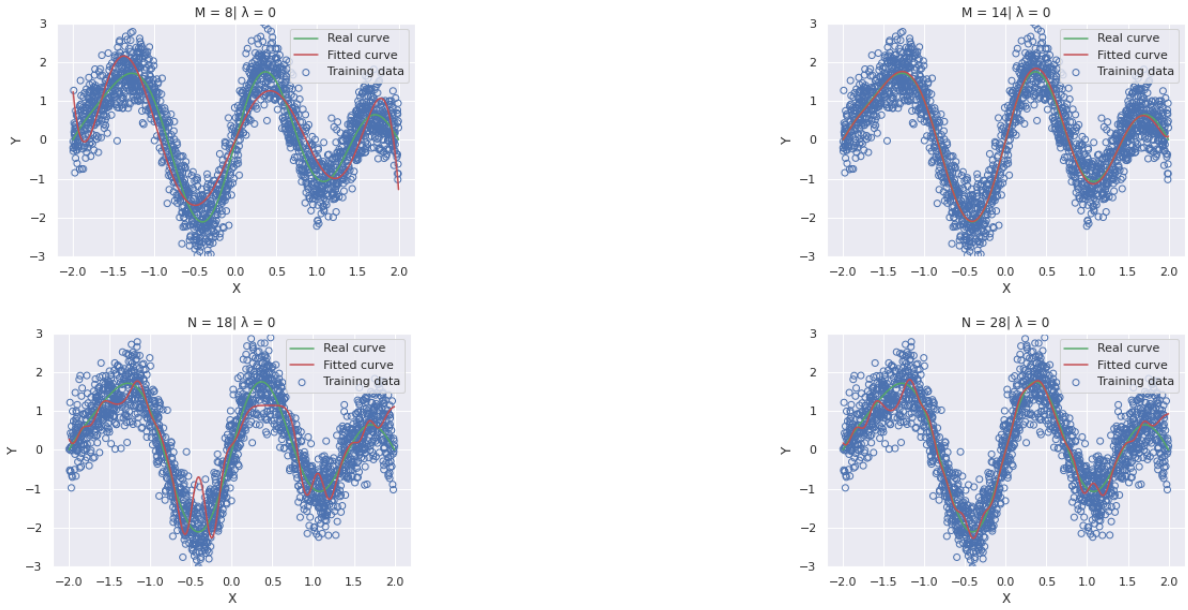


Figure 4: (Top)Plots of the solutions obtained by fitting the $M = 8$-polynomial and the $M = 14$-polynomial to an increased-size dataset containing 1000 data points. Only the $M = 14$-polynomial is capable of perfectly representing the real curve (shown in green). Besides, this plot shows that the $M = 14$-polynomial does not suffer from overfitting when the training data is large. (Bottom) Plots of the solutions obtained by fitting the $N = 18$-radial and the $N = 28$-radial to an increased-size dataset containing 1000 data points. The $N = 28$-radial achieves a better performance than the $N = 18$-radial but it is outperformed by the $M = 14$-polynomial.

require different projections onto a one-dimensional space to be still linearly separable.

In Figure 10, D_1 and D_2 have been generated using the following covariance matrices, respectively

$$S_a = \begin{pmatrix} 20 & 0 \\ 0 & 300 \end{pmatrix}, \quad S_b = \begin{pmatrix} 10 & 0 \\ 0 & 80 \end{pmatrix}$$

In Figure 11, D_1 and D_2 have been generated using the following covariance matrices, respectively

$$S_a = \begin{pmatrix} 15 & 10 \\ 10 & 10 \end{pmatrix}, \quad S_b = \begin{pmatrix} 35 & 30 \\ 30 & 60 \end{pmatrix}$$

3.3 Exercise 3

This exercise consists of finding the rotation θ^* of an initial vector $\boldsymbol{w}_i$ that maximises the Fisher ratio $F(\boldsymbol{w})$. Let

$$\boldsymbol{w}_i(\theta) = R(\theta)\boldsymbol{w}_i = \begin{pmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{pmatrix} \boldsymbol{w}_i$$

be the function that takes a rotation θ as input and returns the vector $\boldsymbol{w}_i$ rotated by an angle equivalent to θ.

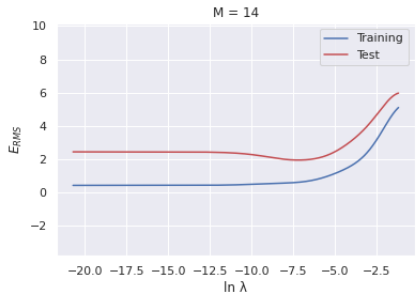
Thus, the optimal vector $\boldsymbol{w}^*$ can be derived from the equation

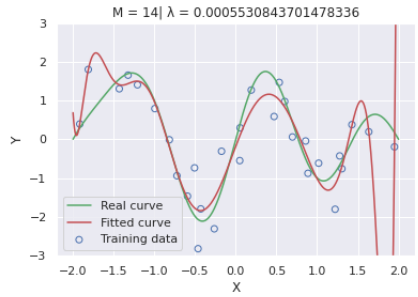
$$\boldsymbol{w}^* = argmax_{\boldsymbol{w}}\ F(\boldsymbol{w}) = argmax_{\theta}\ F(\boldsymbol{w}(\theta))$$

The initial vector $\boldsymbol{w}_i$ has been chosen to be vector $\boldsymbol{w}_2$ from Exercise 1, that is, the vector parallel to the stretching direction of D_1 and D_2. Given that $\boldsymbol{w}_2$ has unit length, the task of this exercise revolves to finding which vector enclosed in a unit circle centred about the origin maximises the Fisher ratio. The approach taken to compute both θ^* and $\boldsymbol{w}^*$ consists of generating 10,000 evenly spaced angles between 0π and π and select among them the one that maximises the Fisher ratio F(w). Using this empirical approach leads to the following result

$$\theta^* = 0.4974649746497465\boldsymbol{\pi}, \quad \boldsymbol{w}^* = (0.9268979, 0.37531358)$$

Figure 12 illustrates the value of the Fisher ratio $F(w)$ given a rotation of the initial vector w_i





	M=14|lnλ=-inf	M=14|lnλ=-10.0	M=14|lnλ=-7.5
w0	-0.138867	-0.155440	-0.182513
w1	6.606789	6.358831	5.554535
w2	-2.791821	-2.233000	-1.574491
w3	-20.604497	-18.601355	-12.473927
w4	10.859870	7.796792	4.396009
w5	26.588995	21.766480	7.403797
w6	-15.870708	-9.729290	-3.135830
w7	-21.864297	-16.889865	-2.207050
w8	11.044562	5.341925	-0.589897
w9	10.922039	8.454816	1.153965
w10	-3.698234	-1.049448	1.610525
w11	-2.800325	-2.222774	-0.495511
w12	0.547738	-0.047092	-0.621011
w13	0.275448	0.224275	0.069050
w14	-0.026075	0.025216	0.072658

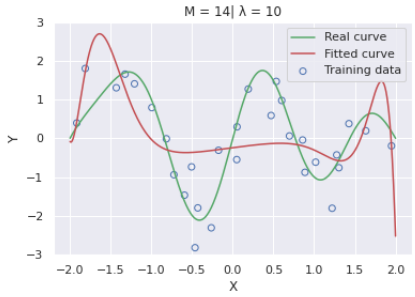


Figure 5: The top-left plot shows the value of the RMS error for both training and test sets against the regularisation parameter for the $M = 14$-polynomial. The optimal regularisation parameter ($e^{-7.5}$) is the point at which the test loss achieves its minimum. The bottom-left plot shows the curve obtained by fitting the $M = 14$-polynomial to the training data with the optimal regularisation parameter. The table in the middle shows the increasingly lower coefficients of the $M = 14$-polynomial as the regularisation parameter increases. The plot on the right shows the $M = 14$-polynomial fitted to the training data using a large regularisation parameter. As it is possible to notice, a large regularisation parameter can lead a model to underfit the data.

4 Data 2: Iris data

In this section, Fisher LDA is used to project a 4-dimensional 3-class dataset onto a 2-dimensional space solving the generalised eigenvalue problem. Thus, Fisher LDA can be generalised to $K > 2$ classes and $D > 1$-dimensional projections. In what follows, three short proofs (not enough space for rigorous mathematical proofs) for three claims are given.

Claim 1: Let D be a T-dimensional dataset containing C classes. Assume that D can be projected onto a T'-dimensional space. Then, the optimal projection matrix is $W = [\boldsymbol{e}_1, \boldsymbol{e}_2, ..., \boldsymbol{e}_{T'}]$ where $\boldsymbol{e}_i$ is the i^{th} eigenvector of the generalised Fisher LDA eigenvector problem. The corresponding eigenvalue of $\boldsymbol{e}_i$ is the i^{th} largest eigenvalue.

Proof: In order to prove the above claim, firstly it will be proved that if we were to project D into a one dimensional space then $W = [\boldsymbol{e}_1]$.

The Fisher ratio is

$$F(\boldsymbol{w}) = \frac{\boldsymbol{w}^T \Sigma_b \boldsymbol{w}}{\boldsymbol{w}^T \Sigma_w \boldsymbol{w}}$$

Differentiating with respect to $\boldsymbol{w}$, $F(\boldsymbol{w})$ is maximized when

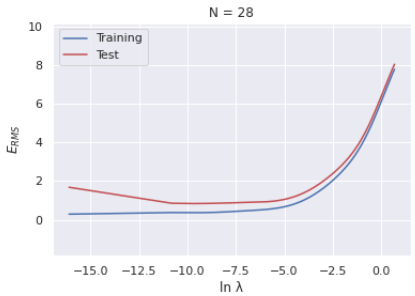
$$(\boldsymbol{w}^T \Sigma_b \boldsymbol{w}) \Sigma_w \boldsymbol{w} = (\boldsymbol{w}^T \Sigma_w \boldsymbol{w}) \Sigma_b \boldsymbol{w}$$
$$\rightarrow \Sigma_b \boldsymbol{w} = \frac{\boldsymbol{w}^T \Sigma_b \boldsymbol{w}}{\boldsymbol{w}^T \Sigma_w \boldsymbol{w}} \Sigma_w \boldsymbol{w}$$
$$\rightarrow \Sigma_b \boldsymbol{w} = F(w) \Sigma_w \boldsymbol{w}$$
$$\rightarrow \Sigma_b \boldsymbol{w} = \lambda \Sigma_w \boldsymbol{w}$$

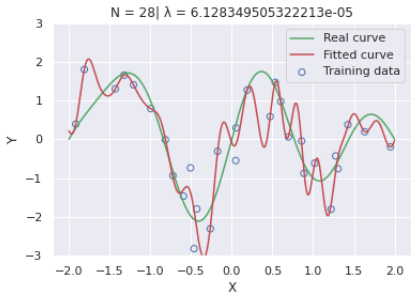
Thus, when projecting onto a one dimensional space, the Fisher ratio is maximised when the eigenvector associated with the largest eigenvalue is chosen. Starting from what has been demonstrated so far, a simple proof by induction can show that Claim 1 is correct.

Claim 2: Let D be a T-dimensional dataset containing C classes. Assume that D is projected onto a vector $\boldsymbol{e}_i$ where $\boldsymbol{e}_i$ is the i^{th} eigenvector of the generalised Fisher LDA eigenvector problem. Then, the Fisher ratio $F(\boldsymbol{e}_i)$ is equal to λ_i where λ_i is the eigenvalue associated with $\boldsymbol{e}_i$.

Proof: The proof comes directly from the proof for Claim 1.

Claim 3: Given a T-dimensional dataset D containing C classes such that $T > D$, D can be projected onto a T'-dimensional space such that $T' \leq C - 1$. Put simply, using Fisher LDA it is only possible to project a dataset onto a space whose dimensionality is at most the number of classes in the dataset minus 1.





	N=28|lnλ=-inf	N=28|lnλ=-12.5	N=28|lnλ=-9.7
w0	1.156371	1.132847	1.121091
w1	-4.770394	-3.900912	-2.454190
w2	-1.598917	-1.301651	-1.302905
w3	-0.348614	-0.304043	-0.294006
w4	-0.188939	-0.180174	-0.176515
w5	0.599384	0.622255	0.634627
w6	23.987410	21.601643	14.624206
w7	8.234522	1.248132	0.903531
w8	9.129290	6.993397	0.795116
w9	-8.080642	-3.301557	-3.070935
w10	1.631976	1.646836	1.653739
w11	1.691338	-0.147238	-1.983416
w12	-669.825602	-36.213801	-3.099257

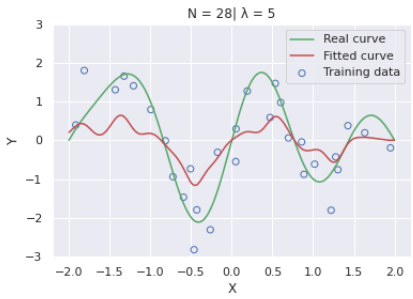


Figure 6: The top-left plot shows the value of the RMS error for both training and test sets against the regularisation parameter for the $N = 28$-radial. The optimal regularisation parameter ($e^{-9.71}$) is the point at which the test loss achieves its minimum. The bottom-left plot shows the curve obtained by fitting the $N = 28$-radial to the training data with the optimal regularisation parameter. The table in the middle shows the increasingly lower coefficients of the $N = 28$-radial as the regularisation parameter increases. The plot on the right shows the $N = 28$-radial fitted to the training data using a large regularisation parameter. As it is possible to notice, a large regularisation parameter can lead a model to underfit the data.

Proof: The generalised eigenvalue problem is

$$\Sigma_b \boldsymbol{w} = \lambda \Sigma_w \boldsymbol{w} \rightarrow \Sigma_w^{-1} \Sigma_b \boldsymbol{w} = \lambda \boldsymbol{w} \tag{1}$$

The rank of the multiplication of two matrices A and B is

$$rank(AB) \leq min(rank(A), rank(B))$$

It is straightforward to prove that the rank of Σ_b is at most $C-1$. Thus, $\Sigma_w^{-1}\Sigma_b$ has at most $C-1$ eigenvectors that do not belong to its null space. In other words, at most $C-1$ eigenvectors can be utilized for the projection process in the Fisher LDA. This comes from the fact that projecting onto a vector associated with a 0 eigenvalue would lead the Fisher ratio along that dimension to be equal to 0 according to Claim 2.

4.1 Exercise 1

According to Claim 2, the optimal direction $\boldsymbol{w}^*$ for projecting the data onto is the first eigenvector that solves the generalised eigenvector problem. The first eigenvector is the eigenvector that is associated with the largest eigenvalue. The eigenvectors and their associated eigenvalues that solve the generalised eigenvector problem for the Iris dataset are approximately (2-decimal approximation):

$$\boldsymbol{e}_1 = (0.12, 0.29, -0.37, -0.41), \quad \lambda_1 = 0.84$$

$$\boldsymbol{e}_2 = (-0.03, 0.36, -0.11, 0.41), \quad \lambda_2 = 0.006$$

$$\boldsymbol{e}_3 = (-0.04, -0.22, -0.29, 0.66), \quad \lambda_3 = 0$$

$$\boldsymbol{e}_4 = (-0.51, 0.34, 0.42, -0.43), \quad \lambda_4 = 0$$

Thus, $\boldsymbol{w}^* = \boldsymbol{e}_1$. Consequently, the Fisher ratio is λ_1 when the dataset is projected onto $\boldsymbol{w}^*$.

It is important to notice that the scipy function eigh did not return exactly 0 eigenvalues for the third and fourth eigenvectors. This is due some rounding errors during the computation. Claim 3 above gives a proof that they need to be exactly 0.

The generalised eigenvector problem has been solved on the training dataset of the Iris dataset which accounts for 80% of the overall dataset. By doing it this way, it is possible to test the performance of Fisher LDA on the test dataset.

4.2 Exercise 2

Figure 13 illustrates the projections of the whole Iris dataset along each of the eigenvectors learnt in the previous exercise.

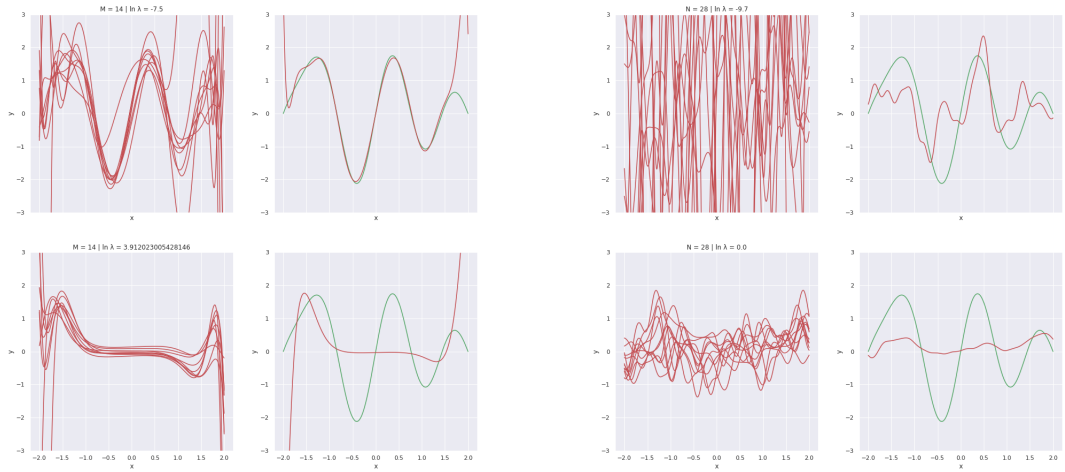


Figure 7: Illustration of the bias-variance tradeoff for the $M = 14$-polynomial(Left) and the $N = 28$-radial(Right). The left column of both plots shows the result of fitting the corresponding model on 1000 different datasets for two different values of the regularisation parameter. The right column of both plots shows the corresponding average of the 1000 fits (red) together with the real function (green). The top row of both plots shows that, given that the optimal regularisation parameter is used, there is large variance between the red curves in the left plot but low bias (good fit between the average model fit and the real curve). When a large value for the regularisation parameter is used (bottom rows), this gives low variance but high bias because the two curves in the right-column plots look very different.

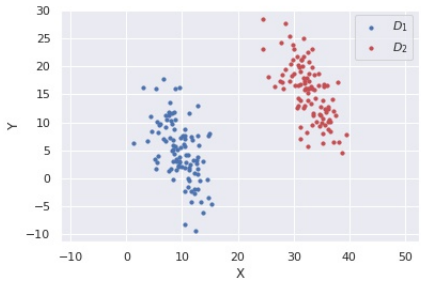


Figure 8: Scatter plot of the generated datasets D_1 and D_2. D_1 and D_2 are clearly linearly separable

4.3 Exercise 3

Figure 14 illustrates the projection of the Iris dataset onto a two-dimensional space. Such a projection is obtained by taking the dot product of each Iris data point with the two eigenvectors corresponding to the two largest generalised eigenvalues.

4.4 Exercise 4 and 5

The Iris dataset has been split into training(80%) and test(20%) datasets. Softmax regression and Fisher LDA have been trained on the training dataset and have achieved both an accuracy of 97.77% on the test dataset. Even though the accuracies achieved are equivalent, Softmax regression and Fisher LDA are quite different as while the former is a probabilistic discriminative model, the latter is a probabilistic generative model. Indeed, while Softmax regression estimates the posterior class probabilities $p(C_k|x)$ using maximum likelihood, Fisher LDA finds the posterior class probabilities $p(C_k|x)$ by infering the class-conditional densities $p(x|C_k)$, the prior class probabilities $p(C_k)$ and then applying the Bayes' Theorem on them. The assumption that the Fisher LDA makes is that the class-conditional densities $p(x|C_k)$ are multivariate Gaussian with class means m_k and a common covariance matrix Σ. The assumption of a common covariance matrix Σ is necessary to obtain a linear score function. Given that I have implemented Fisher LDA myself, I have chosen the class means m_k to be the corresponding class sample means, and the common covariance matrix Σ to be the covariance matrix of the training samples from class 3. The rationale behind this choice is that class 2 and class 3 have a similar covariance matrix and so by choosing one of their covariance matrices, a good approximation of their actual probability densities is achieved. By having good approximations for these densities, it is less likely to misclassify class 2 and class 3 given that, clearly, they are not linearly separable. On the other hand, class 1's probability density is poorly approximated but that is not a problem because it is significantly separated from the other two classes. Indeed, class 1 is never misclassified in the test set. The prior class probabilities $p(C_k)$ have been picked to be $\frac{1}{3}$ each. Classification of the test dataset has been conducted by assigning each sample to the class that achieves the highest linear score. To conclude, for this dataset, Softmax regression and Fisher LDA seem to have a comparable performance. This suggests that it is reasonable to assume that the class samples are drawn

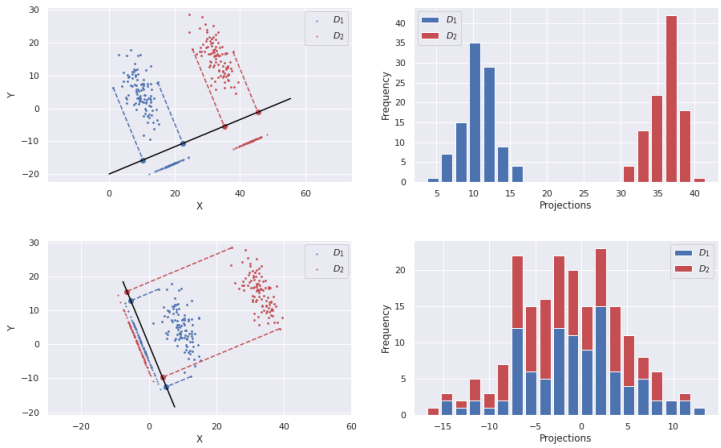


Figure 9: The top-left plot shows the projection of D_1 and D_2 onto a line parallel to $\boldsymbol{w}_1$. The two projected datasets do not overlap as the histograms of their projections in the top-right figure demonstrate. The bottom-left plot shows the projection of D_1 and D_2 onto a line parallel to $\boldsymbol{w}_2$. The bottom-right plot demonstrates that such projections are not linearly separable as the histograms of the projections overlap.

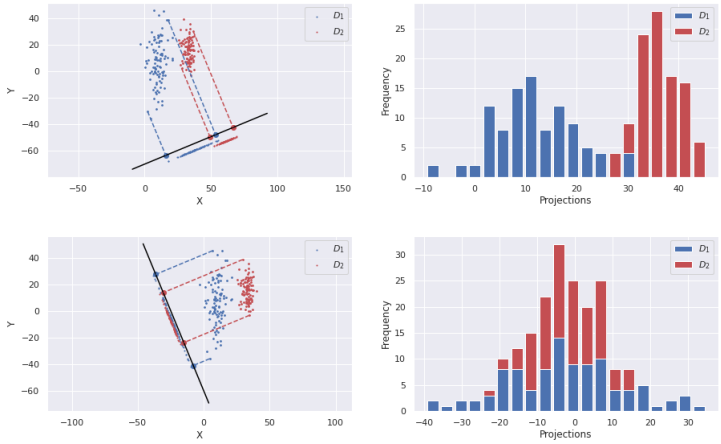


Figure 10: The top plots show the projection of D_1 and D_2 onto a line parallel to $\boldsymbol{w}_1$. While in the previous exercise, projecting onto this line achieved linear separability between the projected datasets, the same does not hold with the newly generated datasets. Indeed, an horizontal line is necessary to obtain non-overlapping projections. The bottom plots show the projection of D_1 and D_2 onto a line parallel to $\boldsymbol{w}_2$. Like in the previous exercise, the projections onto $\boldsymbol{w}_2$ heavily overlap.

from a multivariate Gaussian. If this were not the case, Softmax regression would have outperformed Fisher LDA as it makes no assumption over the distribution of the classes.

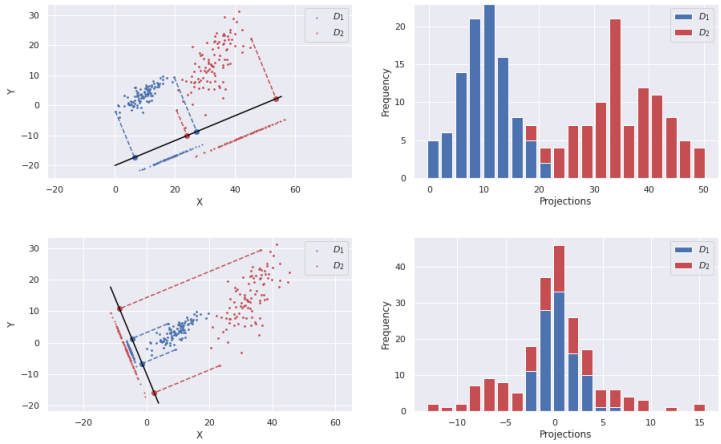


Figure 11: The top plots show the projection of D_1 and D_2 onto a line parallel to $\boldsymbol{w}_1$. While in the previous exercise, projecting onto this line achieved linear separability between the projected datasets, the same does not hold with the newly generated datasets. Indeed, a steeper line line is necessary to obtain non-overlapping projections. The bottom plots show the projection of D_1 and D_2 onto a line parallel to $\boldsymbol{w}_2$. Like in the previous exercise, the projections onto $\boldsymbol{w}_2$ heavily overlap.

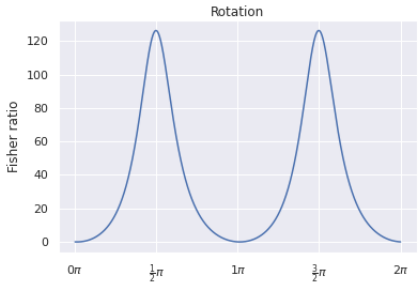


Figure 12: This figure plots the value of the Fisher ratio $F(\boldsymbol{w})$ given a rotation θ of the initial vector $\boldsymbol{w}_i$. The Fisher ratio is maximised when the vector $\boldsymbol{w}_i$ is rotated by an angle of roughly 90°. Therefore, the optimal vector $\boldsymbol{w}^*$ is approximately the vector perpendicular to $\boldsymbol{w}_i$, which in Exercise 1 was called $\boldsymbol{w}_1$. This result explains why the projection of D_1 and D_2 onto $\boldsymbol{w}_1$ in Exercise 1 resulted in a linearly separable dataset. This plot also shows that the worst projection performance is achieved when $\boldsymbol{w}_i$ is not rotated. This result explains why the projection of D_1 and D_2 onto $\boldsymbol{w}_2$ ($\boldsymbol{w}_2 = \boldsymbol{w}_i$) in Exercise 1 resulted in a non linearly separable dataset.

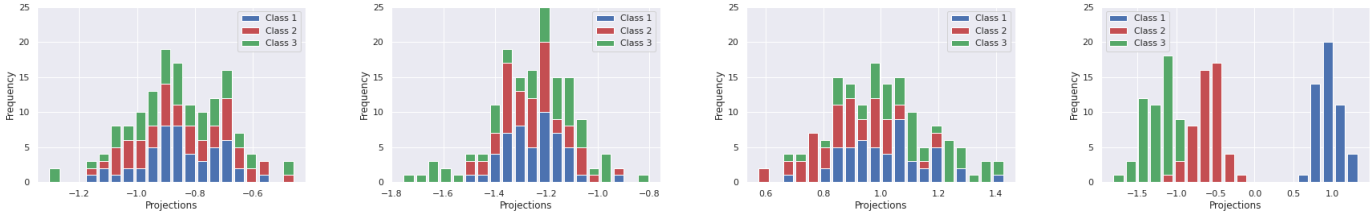


Figure 13: The plots are indexed from left to right. The first and second plots illustrate the projections of the dataset onto $\boldsymbol{e}_4$ and $\boldsymbol{e}_3$, respectively. The three classes heavily overlap, and this reflects the fact that the Fisher ratio for these projections is 0. The third histogram shows the projections onto $\boldsymbol{e}_2$. There is still overlapping between the classes, as confirmed by the low value of λ_2. Finally, the fourth histogram is the projection onto $\boldsymbol{e}_1$. This projection achieves the best performance in terms of class separation. However, as it is possible to notice, Class 2 and Class 3 slightly overlap. Therefore, it is not possible to project the Iris dataset onto a one-dimensional space such that the three classes are linearly separable.

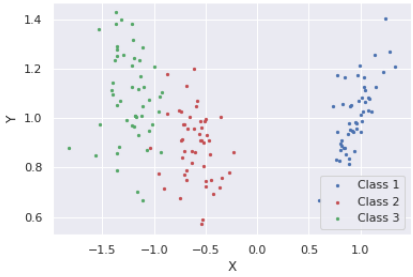


Figure 14: This plot shows the optimal projection of the Iris dataset onto a 2D space. As it is possible to notice, Class 2 and Class 3 overlap. As a consequence, the 2D projection of the Iris dataset is not linearly separable. This result was expected given that Class 2 and Class 3 overlap in the same region when projected onto the optimal eigenvector $\boldsymbol{e}_1$. Projecting also onto $\boldsymbol{e}_2$ creating a 2D projection does not prevent the two classes from overlapping.