COMP0078 - Supervised Learning - Coursework 1

November 10, 2023

COMP0078 Supervised Learning - Coursework 1

Student ID: 18006555

1 PART I

1.1 1.1 Linear Regression

Question 1

We start with a small data set of points $S = \{(1,3),(2,2),(3,0),(4,5)\}$.

By using the polynomial basis $\phi = (\phi_1(x) = 1, \phi_2(x) = x, \cdots, \phi_k(x) = x^{k-1})$ of dimension k (order $k-1$), we can fit the data set S with varying $k = 1,2,3,4$.

We denote the input data $\mathbf{x} = (1,2,3,4)^T$ and the output $\mathbf{y} = (3,2,0,5)^T$, split from the original S data set. We transform $\mathbf{x}$ via the basis functions $(\phi_1, \cdots, \phi_k)$ $(k = 1,2,3,4)$, with our Φ defined as follows:

$$\Phi = \begin{bmatrix} \phi(x_1) \\ \vdots \\ \phi(x_4) \end{bmatrix} = \begin{bmatrix} \phi_1(x_1) & \dots & \phi_k(x_1) \\ \vdots & \ddots & \vdots \\ \phi_1(x_m) & \dots & \phi_k(x_m) \end{bmatrix} = \begin{bmatrix} 1 & x_1 & \dots & {x_1}^{k-1} \\ 1 & x_2 & \dots & {x_2}^{k-1} \\ 1 & x_3 & \dots & {x_3}^{k-1} \\ 1 & x_4 & \dots & {x_4}^{k-1} \end{bmatrix}$$

where $m = 4$ is the number of data points.

We perform linear regression on the transformed data set to find a k-dimensional vector $\mathbf{w}^{(\mathbf{k})} = (w_1, \cdots, w_k)$ which minimises:

$$SSE = (\Phi \mathbf{w}^{(k)} - y)^T (\Phi \mathbf{w}^{(k)} - y)$$

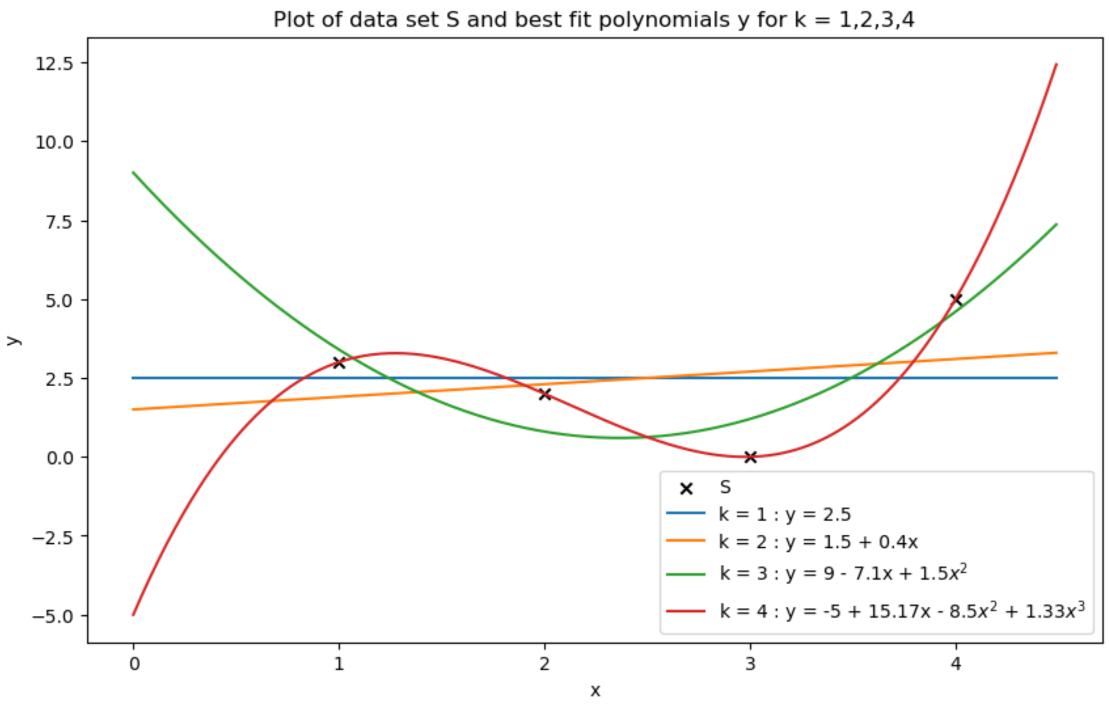
From lectures, we know $\mathbf{w}^{(k)} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{y}$, which utilises the pseudo-inverse of Φ, since it may not have a direct inverse as it may not be square.

Q1 Part (a)

Computing $\mathbf{w}^{(k)}$ for $k = 1,2,3,4$, we can plot the corresponding polynomials $y^{(k)} = \mathbf{w}^{(k)} \phi$ to fit over S on the same plot:

[6]:

```
```



Q1 Part (b)

The equations for the curves shown above are as follows (to 2 decimal places):

$$
\begin{aligned}
y^{(0)} &= 2.5 \\
y^{(1)} &= 1.5 + 0.4x \\
y^{(2)} &= 9 - 7.1x + 1.5x^2 \\
y^{(3)} &= -5 + 15.17x - 8.5x^2 + 1.33x^3
\end{aligned}
$$

Q1 Part (c)

The mean square errors ($MSE = \frac{SSE}{m}$) of each fitted curve $k = 1, 2, 3, 4$ (with $m = 4$), are as follows:

$$
\begin{aligned}
MSE^{(1)} &= 3.25 \\
MSE^{(2)} &= 3.05 \\
MSE^{(3)} &= 0.8 \\
MSE^{(4)} &= 3.6 \times 10^{-27} \approx 0
\end{aligned}
$$

Note: This uses the formula for SSE from above.

Question 2

Q2 Part (a)

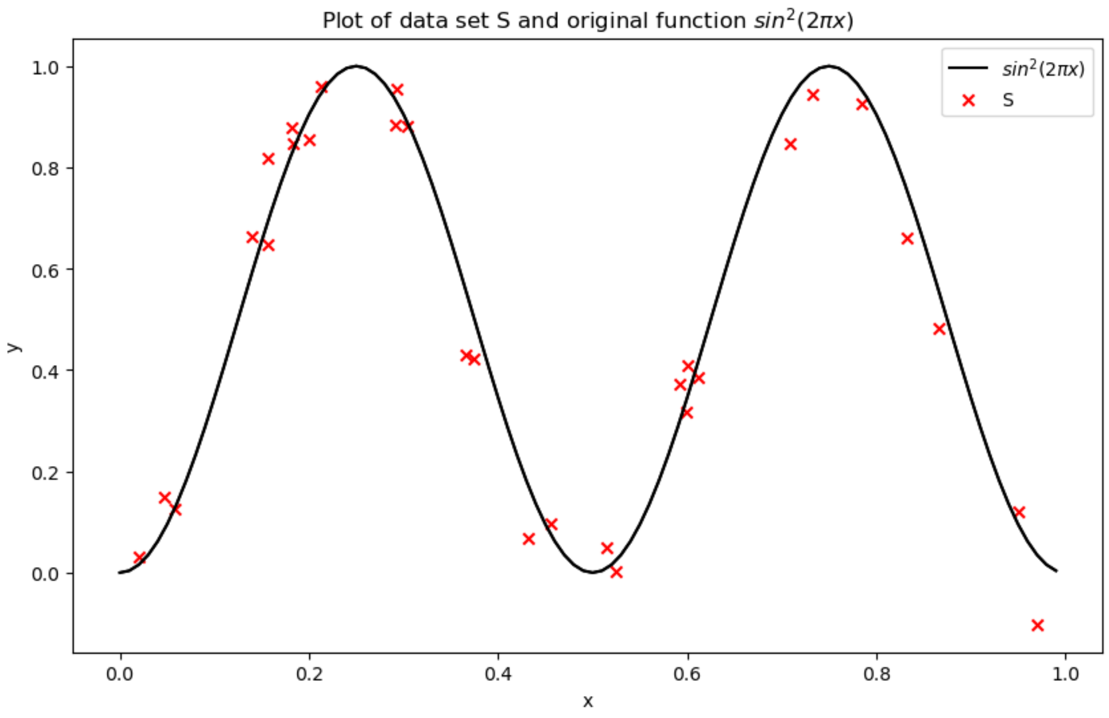
We start with $g_\sigma(x) = \sin^2(2\pi x) + \epsilon$, with $\epsilon \sim N(0, \sigma^2)$.

We then aim to create our sample training data set $S_{(0.07,30)} = \{(x_i, g_\sigma(x_i)\}$ for $i = (1, \cdots, 30)$ by using random x_i and our function g_σ.

Q2a (i)

The below plot shows $sin^2(2\pi x)$ from $0 \leq x < 1$, alongside the scatter plot of the points of $S_{(0.07,30)}$ generated by $g_\sigma(x)$.

[12]:

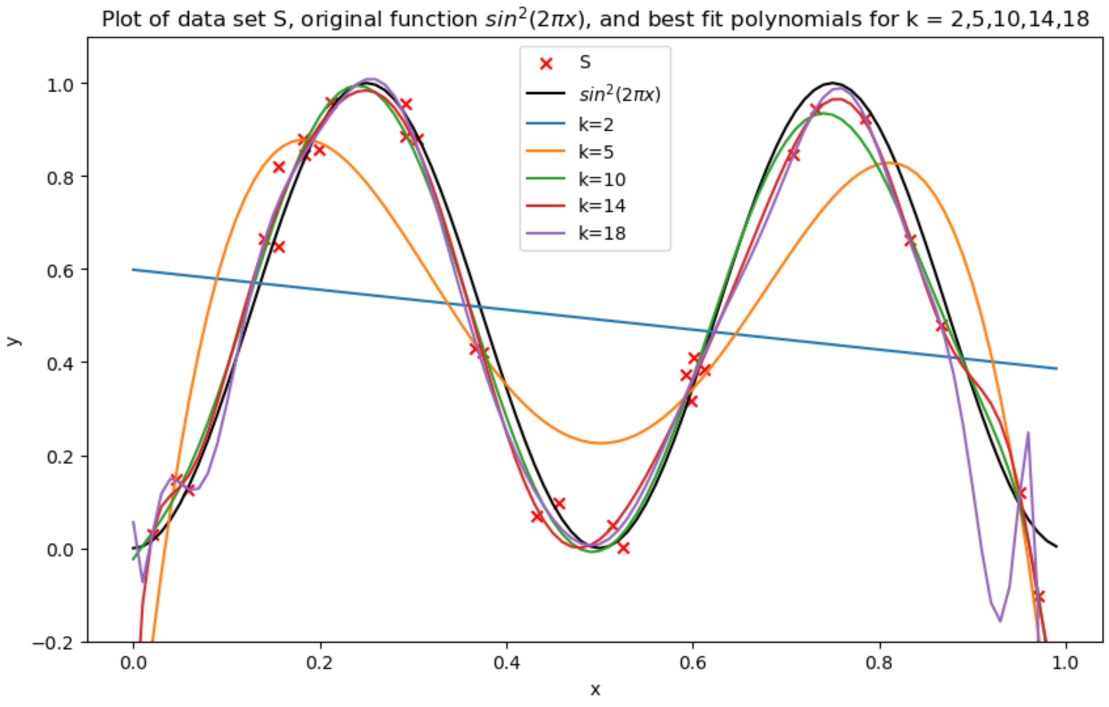


Q2a (ii)

We now aim to fit this data set with polynomial bases of dimension $k = 2, 5, 10, 14, 18$, and plot the final polynomials on the same plot.

Below we identify the relevant weight coefficients w for each dimension k.

[14]:



We observe that as k increases, the curve fits the data set S better - almost too accurately, with $k = 18$ almost over-compensating to fit the points (overfitting), rather than resembling the $sin^2(2\pi x)$ curve which the polynomial was aiming to generalise.

Q2 Part (b)

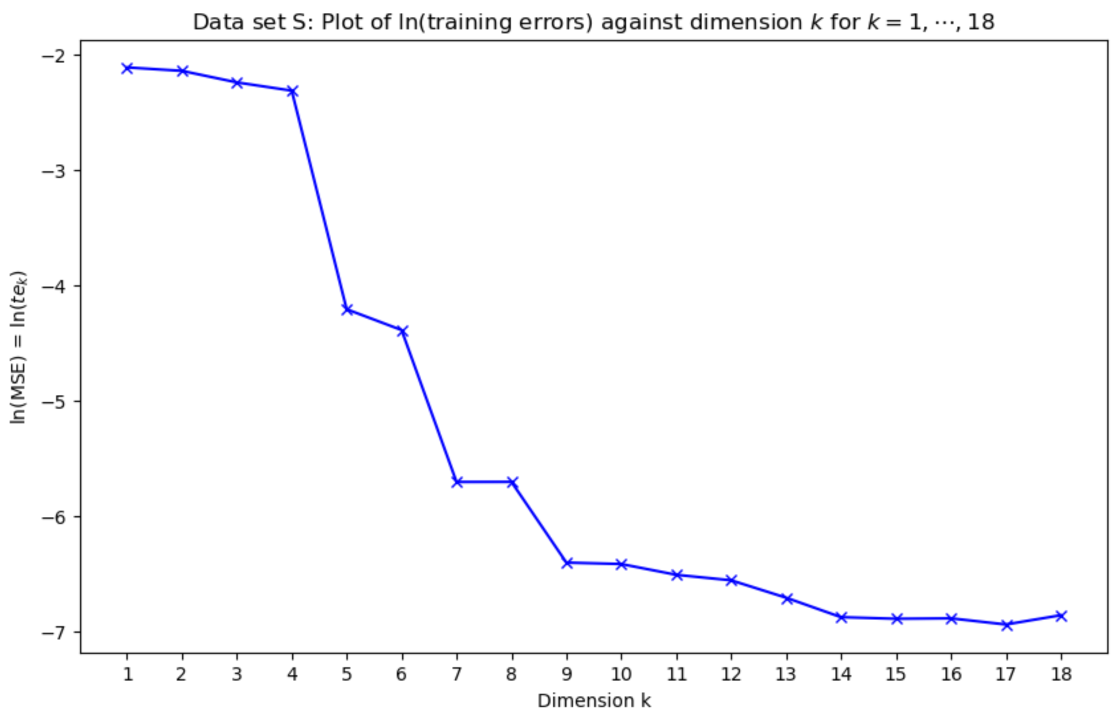
We now aim to evaluate the training error $te_k(S) = MSE$ of these best fit polynomials.

The below plot shows the natural log of the training error (MSE) against the dimension $k = 1, \cdots, 18$, with the polynomial basis:

[16]:

```

```

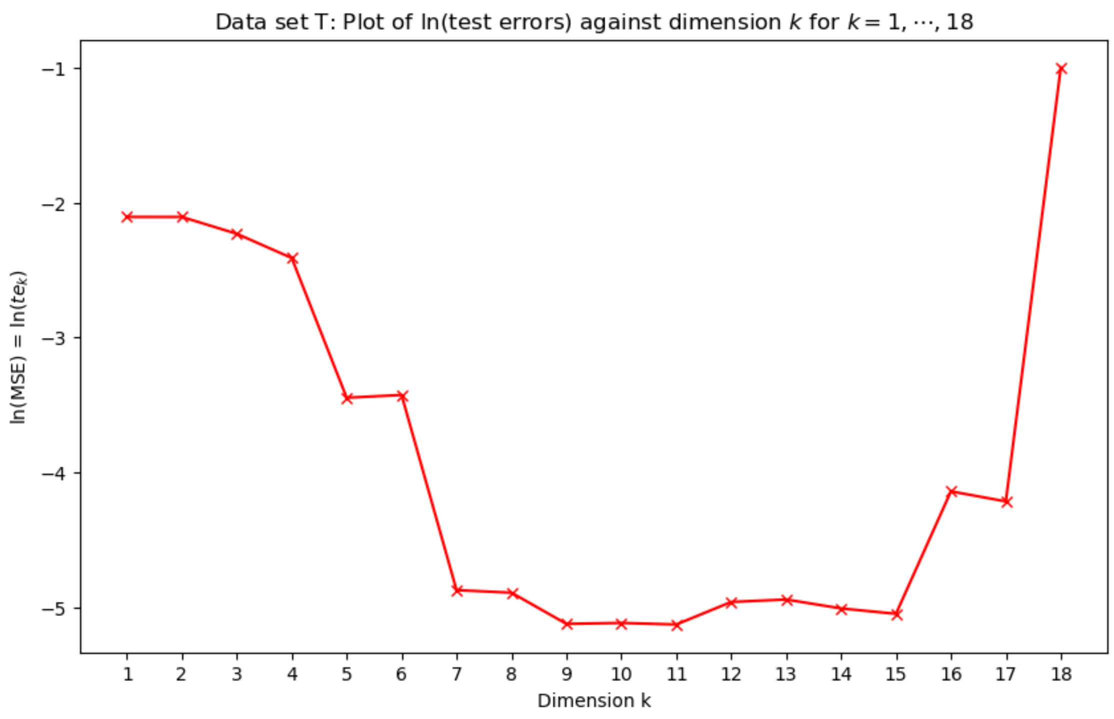


Q2 Part (c)

We now generate 1000 test data points T via the same random generation as S before i.e. $T_{(0.07,1000)} = \{(x_i, g_\sigma(x_i)\}$ for $i = (1, \cdots, 1000)$.

Now we repeat the calculation of our errors, but instead now on the test data using the polynomials generated from the training data. Unlike the training error, this is not a decreasing function and indicates that the higher dimensional polynomials (in particular, $k = 16, 17, 18$) are subject to overfitting with this particular data set.

```
[19]:
```

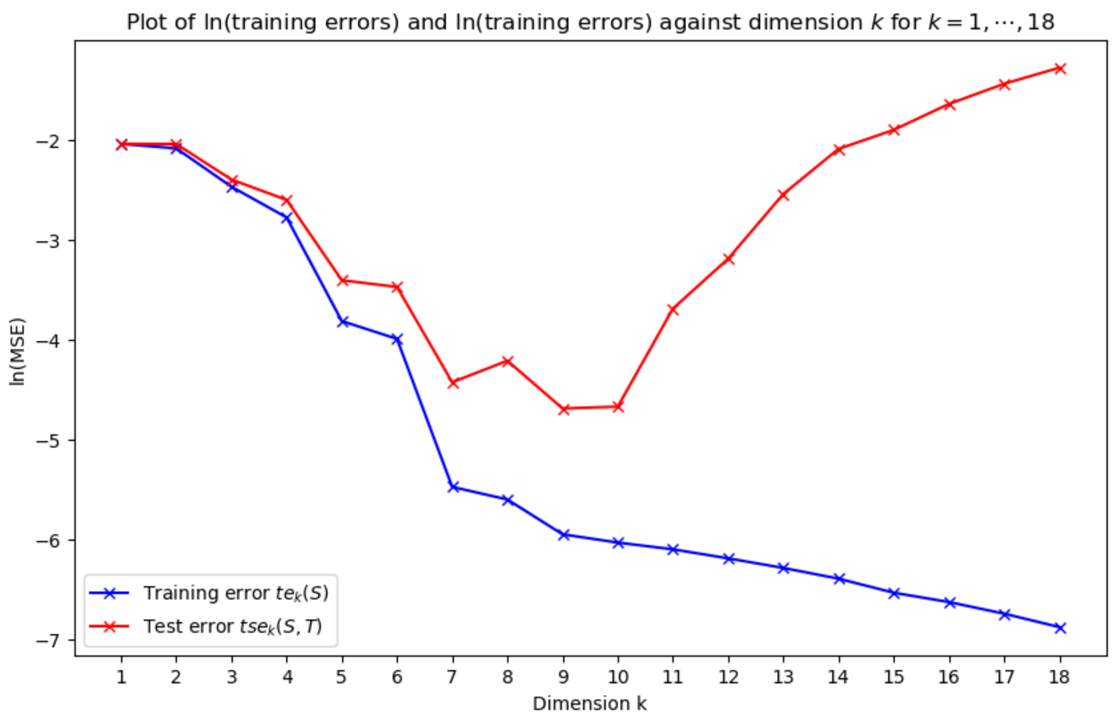


Q2 Part (d)

We now repeat the above methods and average over 100 trials of the above experiments. This aims to negate any impact from the particular data set chosen in one single run, and isolates the impact of overfitting from the higher degree polynomials.

The below plot shows the natural log of both the training and test data's MSE, against the dimension $k = 1, \cdots, 18$. As expected, we see the training error as a decreasing function, whereas the test error appears to noticeably stabilise from $k = 7$ onwards, and increase from $k = 10$, suggesting that $k = 7$ is the optimal dimension for the problem to minimise the MSE, whilst also keeping the model's ability to generalise and reduce overfitting by keeping the dimension low.

[21]:



Question 3

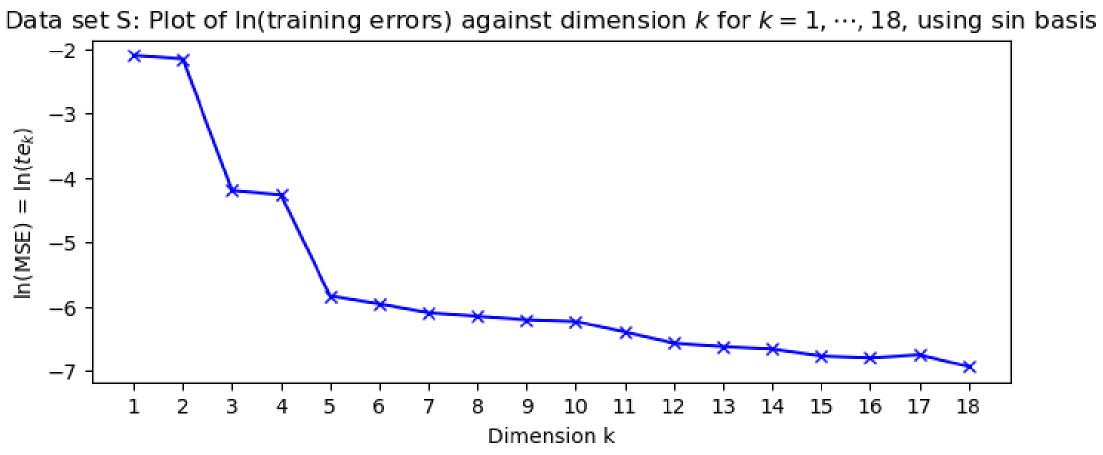
Now instead of the polynomial basis from before, we aim to use the following basis:

$$\{\sin(1\pi x),\ \sin(2\pi x),\ \sin(3\pi x),\ \cdots,\ \sin(k\pi x)\}$$

We repeat parts (b) and (c) with the above basis.

The below plot shows the natural log of the training error (MSE) against the dimension $k = 1, \cdots, 18$, with the sin basis:

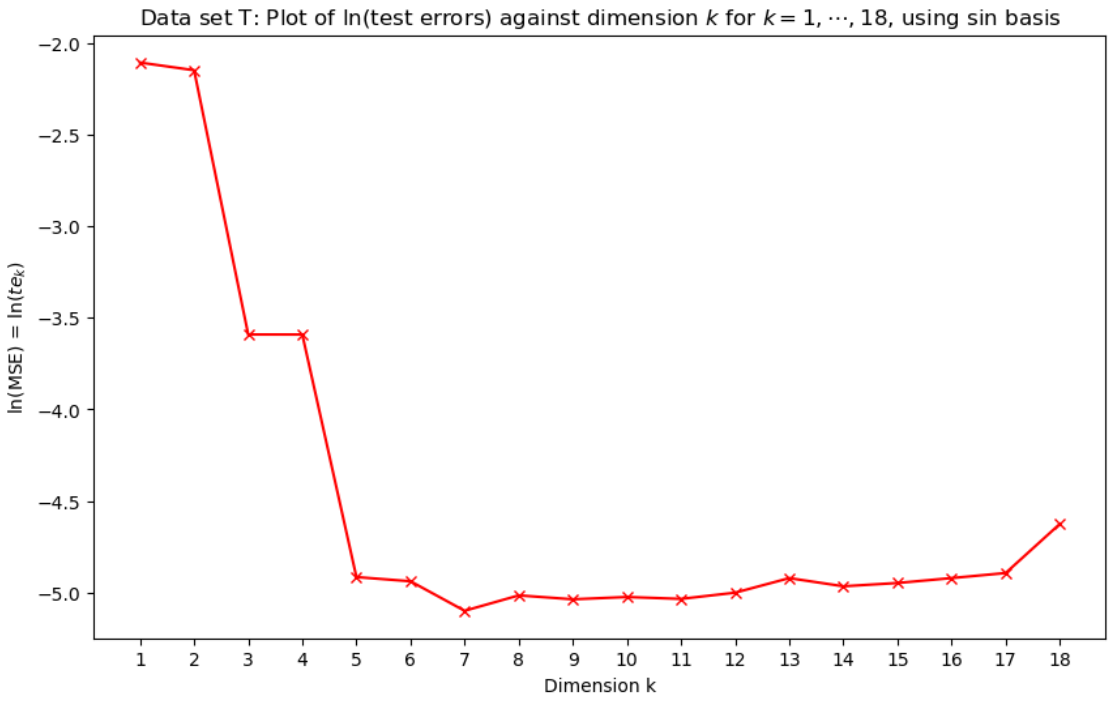
[27]:



Now we compute the test error using the sin basis generated from the training data.

We generate our test data again, with 1000 entries. The below plot shows the natural log of the test MSE, against the dimension $k = 1, \cdots, 18$. We observe that the test error begins to noticeably increase towards larger values of k, for this particular data set.

```
[29]:
```

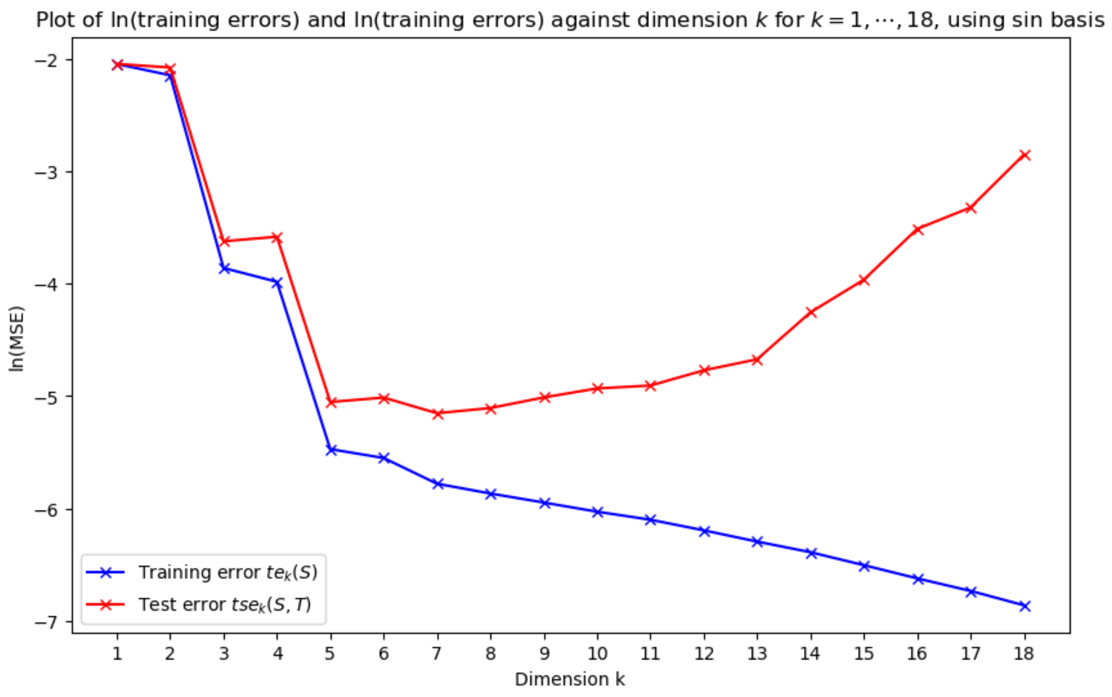


Now, we repeat the experiment 100 times and average over all trials.

The below plot shows the natural log of both training and test MSE against the dimension $k = 1, \cdots, 18$, averaged over 100 randomised data sets. We observe that, similar to the polynomial basis, the sin basis also exhibits overfitting as k increases too far.

As an aside, one may argue that the optimal value of k however is at $k = 5$, which balances minimising the MSE and also overfitting. Since this is a lower complexity than the polynomial basis, this may suggest the sin basis as a more suitable basis for fitting the original curve. Perhaps this is not surprising, as the original curve was sinusoidal.

```
[31]:
```



1.2 Filtered Boston housing and kernels

Question 4

Q4 Part (a)

First, we set our training ratio as 2/3, prior to splitting the Boston housing data set. We aim to perform a naive linear regression (a constant function) over the data set.

Looping over 20 runs, we identify the MSE of the training and test data sets and note them below:

Model	Train MSE	Test MSE
Naive Regression	84.542963 ± 5.389002	84.470437 ± 10.755627

Q4 Part (b)

The constant function in part (a) is simply the mean of the y_{train} data, for a particular run. It is the bias term on its own, with no dependence on values of x_i from the original data set.

Q4 Part (c)

Now we perform a linear regression with a single attribute/feature x_i, for $i = 1, \cdots, 12$. We augment the design matrix with an appended column vector $\mathbf{1}$, to act as the bias term.

Below we see the MSE of the training and test data sets, averaged over 20 runs with random 2/3 training splits, for linear regression with single attributes and a bias term.

Method	MSE Train	MSE Train
CRIM	71.249285 ± 4.894360	73.912137 ± 10.400258
ZN	74.193390 ± 4.343447	72.383101 ± 8.770823
INDUS	64.915538 ± 4.346531	64.559712 ± 8.850307
CHAS	82.016892 ± 5.122321	82.193061 ± 10.622458
NOX	69.475862 ± 4.445054	68.482824 ± 8.976575
RM	43.357060 ± 3.017401	44.461120 ± 5.905816
AGE	72.685168 ± 4.813339	72.258942 ± 9.676213
DIS	79.438796 ± 5.261917	78.976766 ± 10.544300
RAD	71.928464 ± 4.883926	73.079478 ± 9.851037
TAX	65.587031 ± 4.484588	66.968100 ± 9.016871
PTRATIO	62.350071 ± 3.989596	63.832678 ± 8.089564
LSTAT	38.698006 ± 2.333831	38.504028 ± 4.631252

Q4 Part (d)

Now we aim to implement Linear regression using all attributes.

Method	MSE Train	MSE Train
All Attributes	22.279921 ± 1.649453	23.975248 ± 3.585723

Evidently, the MSE for linear regression on all attributes significantly outperforms the results from part (c) where we only trained on an individual attribute.

1.3 Kernelised Ridge Regression

Question 5

We aim to perform kernel ridge regression (KRR) on the data set using the following Gaussian kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = exp\left(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}\right)$$

We set out our parameters γ and σ to prepare a grid search on the optimal pairing to minimise the training MSE, with the γ values being $[2^{-40}, 2^{-39}, \cdots, 2^{-26}]$ and the σ values being $[2^7, 2^{7.5}, \cdots, 2^{13}]$

Q5 Part (a)

From this particular run, the γ and σ which performed the best are:

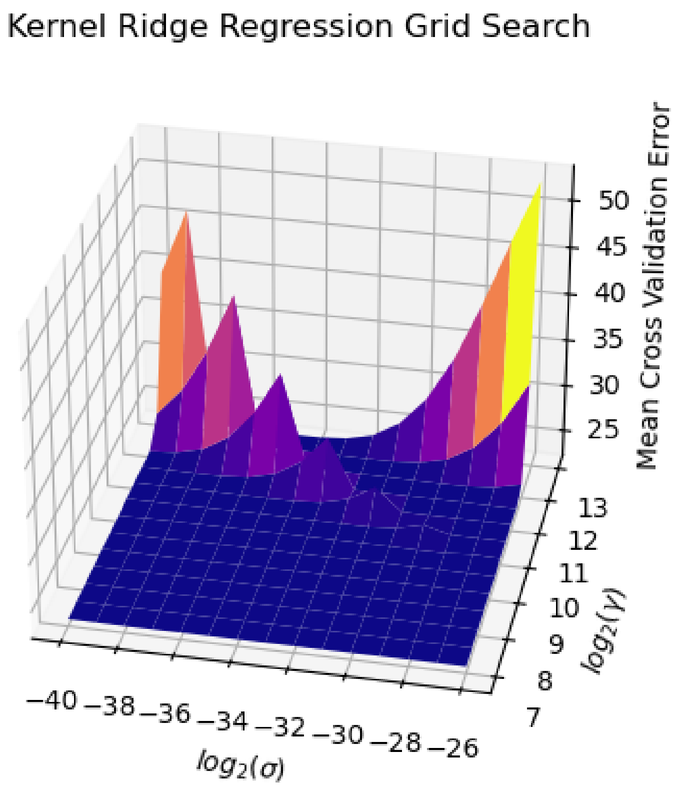
γ^*	σ^*
2^{-35}	2^{13}

Q5 Part (b)

The following 3D plot shows the mean cross validation error as a function of γ and σ from the parameter grid search.

It is to be noted that the minimum point $(\gamma^*, \sigma^*) = (2^{-35}, 2^{13})$ is not an obvious minimum point from the plot, as the surface appears somewhat uniform throughout a lot of the grid. On a different seed, the optimal parameters may change, however the optimal MSE should not differ too much.

[54]:



Q5 Part (c)

Now we aim to retrain our model with the optimal parameters $(\gamma^*, \sigma^*) = (2^{-35}, 2^{13})$, to compute the MSE of the training and test data sets.

From this particular run, the overall training and test MSE are as follows:

Train MSE	Test MSE
0.63859	22.25467

Q5 Part (d)

We now aim to repeat Q5 parts (a) and (c) for Kernel Ridge Regression over 20 random splits of train and test data:

Method	MSE Train	MSE Train
Kernel Ridge Regression	0.742936 ± 0.950861	20.636218 ± 3.277994

Below we show the full summary table, displaying the training and test MSE of all methods used across Q4 and Q5:

Method	MSE Train	MSE Train
Naive Regression	84.542963 ± 5.389002	84.470437 ± 10.755628
CRIM	71.249285 ± 4.894360	73.912137 ± 10.400258
ZN	74.193390 ± 4.343447	72.383101 ± 8.770823
INDUS	64.915538 ± 4.346531	64.559712 ± 8.850307
CHAS	82.016892 ± 5.122321	82.193061 ± 10.622458
NOX	69.475862 ± 4.445054	68.482824 ± 8.976575
RM	43.357060 ± 3.017401	44.461120 ± 5.905816
AGE	72.685168 ± 4.813339	72.258942 ± 9.676213
DIS	79.438796 ± 5.261917	78.976766 ± 10.544300
RAD	71.928464 ± 4.883926	73.079478 ± 9.851037
TAX	65.587031 ± 4.484588	66.968100 ± 9.016871
PTRATIO	62.350071 ± 3.989596	63.832678 ± 8.089564
LSTAT	38.698006 ± 2.333831	38.504028 ± 4.631252
All Attributes	22.279921 ± 1.649453	23.975248 ± 3.585723
Kernel Ridge Regression	0.742936 ± 0.950861	20.636218 ± 3.277994

2 PART II

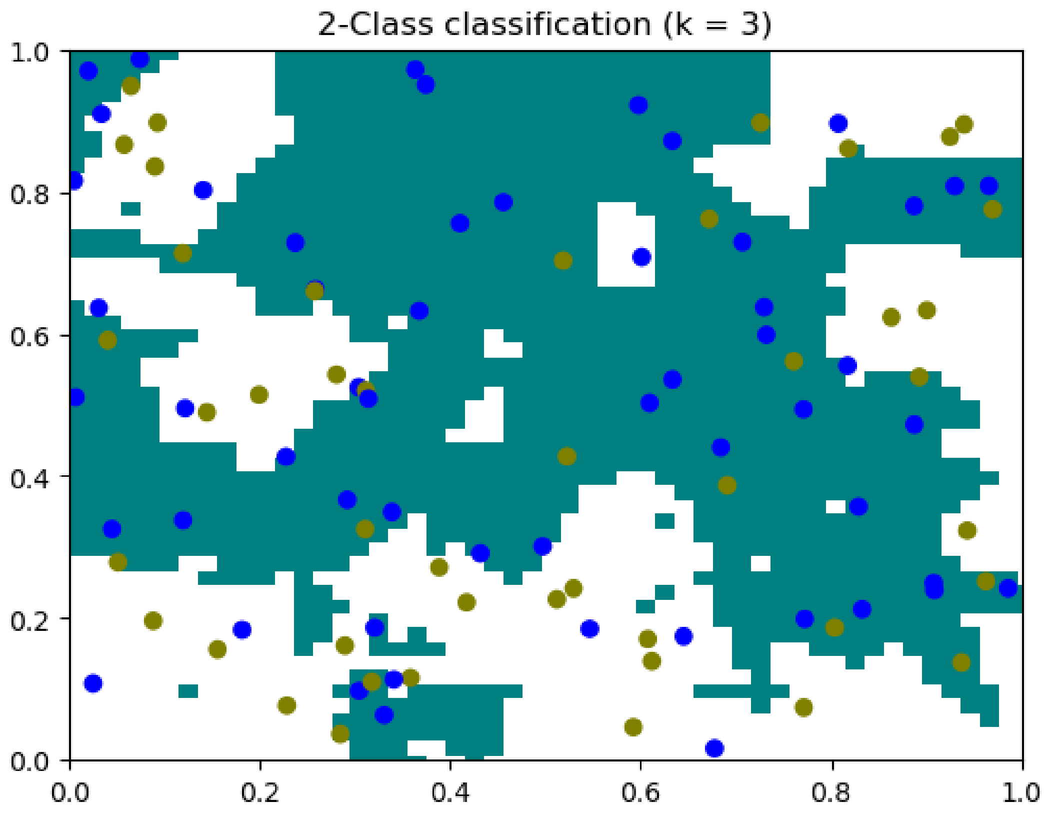
2.1 k-Nearest Neighbours

2.2 Generating the data

Question 6

The below plot shows a visualisation of $h_{S,v}$ similar to Figure 1, with $|S| = 100$ and $v = 3$. Here, the white mapping is to 0 and the teal/turquoise mapping is to 1, with corresponding centers as olive/green and blue.

[60]:



2.3 Estimated generalisation error of k-NN as a function of k

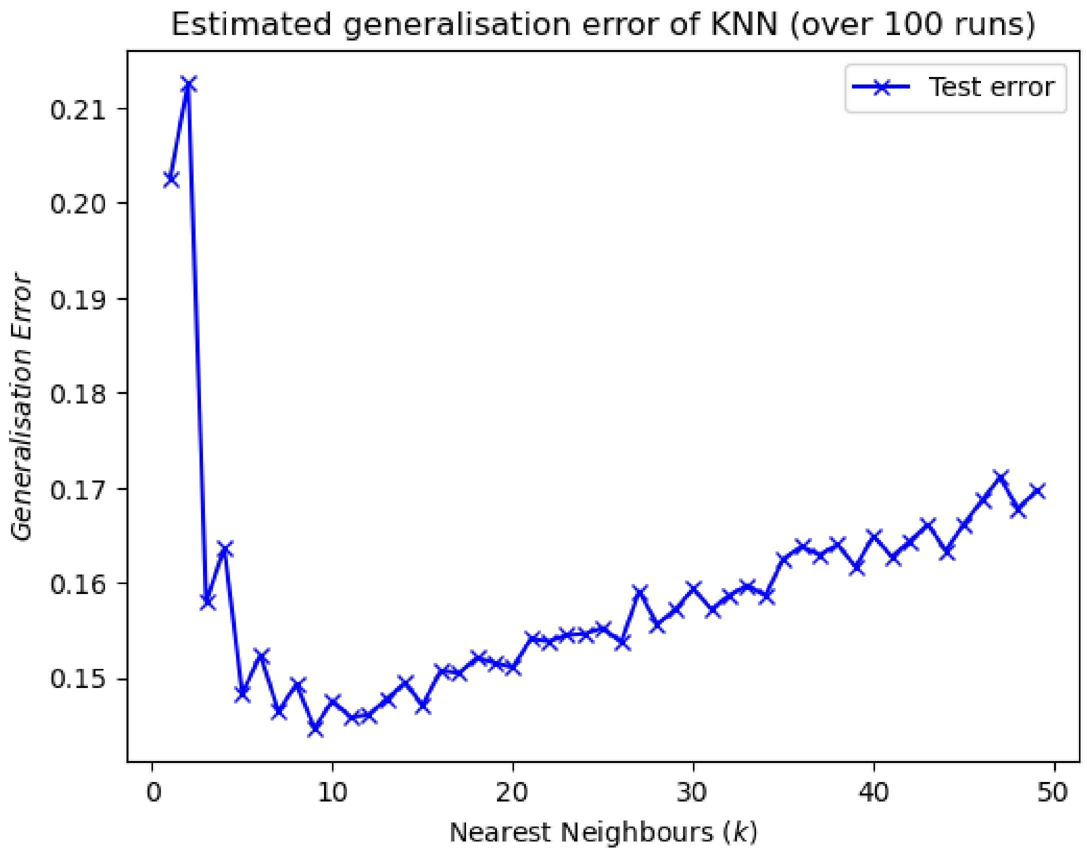
Question 7

Q7 Part (a)

The below plot shows the visualisation of Protocol A from Figure 2:

[65]:

```
```



Q7 Part (b)

We notice an interesting observation that the plot resembles an 'elbow' or 'U-shaped' curve. From $k = 1$ to $k = 5$, the generalisation error starts at a maximum and decreases rapidly to a minimum. From $k = 5$ to $k = 11$, the error appears to stabilise somewhat, with the actual minimum point at $k = 9$, however this does not appear to be a robust minimum and could change with a different seed. From $k = 11$ onwards, we see a gradual yet 'spiky' increase to $k = 50$ and beyond.

For small k (and hence large $\frac{m}{k}$), the error is large due to the model considering fewer neighbors, resulting in a complex, sensitive decision boundary that closely fits the training data, leading to overfitting with high variance and low bias.

For large k (small $\frac{m}{k}$), the error is also high as the model considers more neighbors, creating a less complex decision boundary that struggles to capture underlying patterns, resulting in underfitting with low variance but high bias due to the 'Bias-Variance' tradeoff.

In between, there is a 'sweet spot' for an optimal k, at the minimum of the curve. However, even with a large $m = 4000$ and 100 iterations, there is still substantial noise at the minimum which still makes it difficult to identify an exact optimal k.

Another interesting observation includes that the visualisation appears to suggest a higher error for even values of k, compared to odd, especially at low k. This may be because even values of k

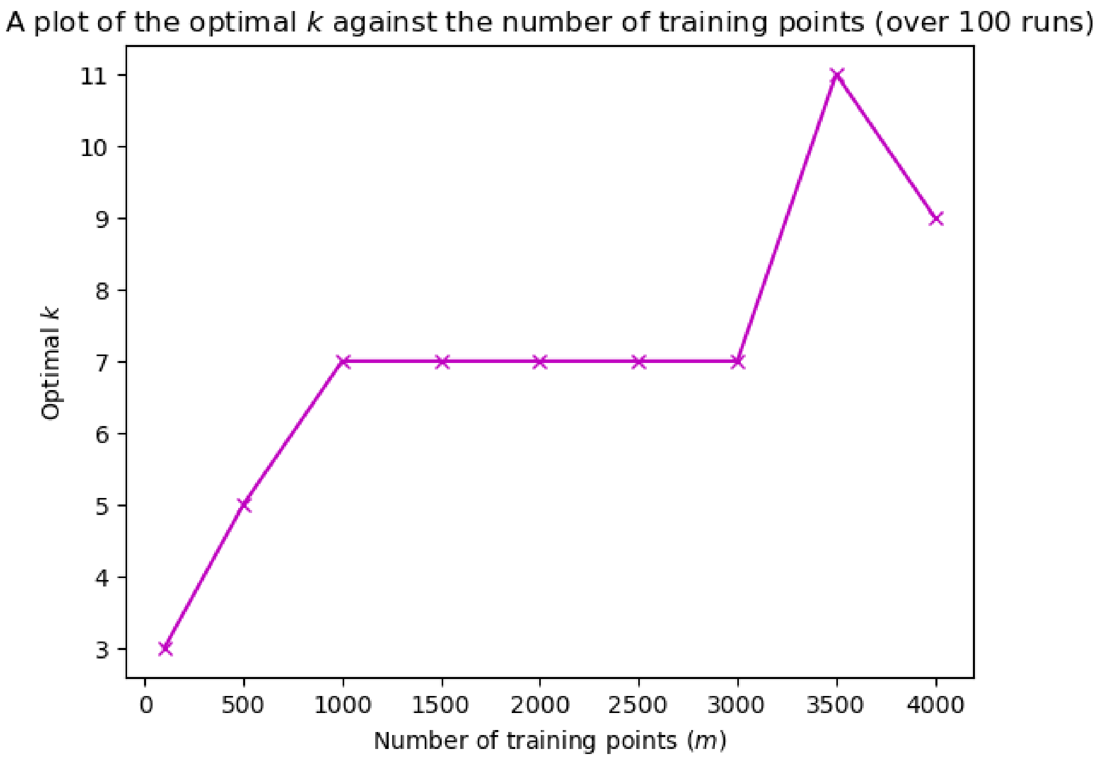
are prone to voting ties, especially with a sensitive decision boundary at lower ks. These ties are resolved by predicting labels uniformly at random, which may lead to a higher generalisation error at these even values of k.

Question 8

Q8 Part (a)

The below plot shows a visualisation of Protocol B, plotting the number of training points m against the optimal k.

[69]:



Q8 Part (b)

The above figure indicates that as the number of training points m increases, the optimal k also increases, but at a decreasing rate and begins to converge to a consistent overall k.

This can be explained by the bias-variance tradeoff, where a small m indicates that the model is sensitive to individual training points (overfitting), resulting in a smaller optimal k. As the training set size increases, there is more data to inhabit trends and patterns within the data, requiring a larger k for better generalisation to capture this complexity. The curve eventually would plateau as eventually more data does not improve the model as much, leading to smaller increases in optimal k.

The curve is still suspect to high variation as displayed from the plot in Q7a, as the ‘sweet spot’ is still not robust and most values of k from 5 to 11 appear to produce similar error values (even at $m = 4000$).

Theoretically, one can show that $\epsilon(k(m)\text{-NN}) \to \epsilon(f^*)$ as $m \to \infty$, provided that $k(m) \to \infty$ and $\frac{k(m)}{m} \to 0$, where $\epsilon(f^*)$ is the ‘true error’. A well-known heuristic for $k(m)$ is $\sqrt{m}$ or even $\ln m$, however this depends on the nature of the dataset provided and the influence of noise, hence is hard to pinpoint.

3 PART III

3.1 Questions

Question 9

Q9 Part (a)

Given the function $K_c(\mathbf{x}, \mathbf{z}) := c + \sum_{k=1}^{n} x_k z_k$ where $\mathbf{x}, \mathbf{z} \in \mathbb{R}^n$, we aim to find what values of $c \in \mathbb{R}$ is K_c a positive semidefinite kernel.

Claim: $c \geq 0$

Proof: We need to show:

(1) Symmetry: $K_c(x, y) = K_c(y, x)$, for all x, y in the domain

(2) Positive Semidefinite Matrix: $\mathbf{v}^T G \mathbf{v} \geq 0$, for any $\mathbf{v} \in \mathbb{R}^n$ and G is the matrix such that $G_{ij} = K_c(\mathbf{x_i}, \mathbf{x_j})$, for all $x_1, x_2, \cdots, x_n$.

For (1), we can show symmetry as follows:

$$\begin{aligned} K_c(\mathbf{x}, \mathbf{z}) &= c + \sum_{i=1}^{n} x_i z_i \\ &= c + \sum_{i=1}^{n} z_i x_i \\ &= K_c(\mathbf{z}, \mathbf{x}) \end{aligned}$$

For (2), we can show this as follows:

First consider $c \geq 0$ as true, we will now show (2) to be true:

$$\begin{aligned} \mathbf{v}^T G \mathbf{v} &= \sum_{i,j}^{n} v_i K_c(\mathbf{x_i}, \mathbf{x_j}) v_j \\ &= \sum_{i,j}^{n} v_i v_j (c + \sum_{k=1}^{n} x_{i,k} x_{j,k}) \\ &= c \sum_{i,j}^{n} v_i v_j + \sum_{i,j}^{n} \sum_{k=1}^{n} v_i x_{i,k} x_{j,k} v_j \\ &= c||\mathbf{v}||^2 + ||\sum_{k=1}^{n} v_k \mathbf{x_k}||^2 \end{aligned}$$

The second term is positive as it is a square (as $\mathbf{a} \neq 0$), and the first term is non-negative if and only if $c \geq 0$, hence the RHS is > 0.

Secondly we consider (2) as true and show $c \geq 0$ to be true, by contradiction:

Assume $c < 0$. Consider $K_c(\mathbf{0}, \mathbf{0}) = c < 0$ and $\mathbf{v} = \mathbf{1}$, which implies $\mathbf{v}^T G\mathbf{v}$ is not positive and (2) does not hold for all values. Hence there is a contradiction and $c \geq 0$.

Q9 Part (b)

The parameter c within this kernel function, acts as a 'bias' term for linear regression (least squares). It has a regularisation effect, where a larger c can prevent overfitting by penalising large coefficients and making the model less sensitive to individual data points, whereas a smaller c allows a more complex decision boundary and could also cause stability issues within the solution when computing a matrix inverse.

Question 10

Given the Gaussian kernel $K_\beta(\mathbf{x}, \mathbf{t}) = \exp(-\beta||\mathbf{x} - \mathbf{t}||^2)$, we aim to understand how the parameter β will enable the trained linear classifier to simulate a 1-NN classifier, where the label of the nearest data point is assigned to the given test point.

Our classifier takes the form $sign(f(\mathbf{t}))$, with $f(\mathbf{t}) = \sum_{i=1}^{m} \alpha_i \exp(-\beta||\mathbf{x}-\mathbf{t}||^2)$. To simulate a 1-NN classifier, we want $f(\mathbf{t})$ to be influenced by the training point $\mathbf{x_{min}}$ which is closest to $\mathbf{t}$, and not to be influenced by any other training points.

Consider the limit as $\beta \to \infty$. The exponential term $\exp(-\beta||\mathbf{x} - \mathbf{t}||^2)$ will approach zero when $\mathbf{x} \neq \mathbf{t}$, and the training point $\mathbf{x_{min}}$ which is closest to $\mathbf{t}$ will have the largest contribution. The value of β indicates the 'strength' of the isolation of the closest training point x_i, making the Gaussian kernel highly peaked, which is similar to the Dirac delta function ($\delta(x - y) = \infty$ when $x = y$ and 0 otherwise) by its spike-like behaviour.

Therefore as $\beta \to \infty$, the classifier $f(\mathbf{t}) \to \alpha_{min} \exp(-\beta||\mathbf{x_{min}} - \mathbf{t}||^2)$, since all other terms will tend to zero. This will mimic the nature of a 1-Nearest Neighbour classifier.

Question 11

Q11 Part (a)

We first show that $\mathcal{E}_{\rho_i}(f_i) = 0$. We observe that:

$$\begin{aligned}\mathcal{E}_{\rho_i}(f_i) &= \int_{\mathcal{C}\times\mathcal{Y}} \mathbf{1}\{f_i(x) \neq y\} d\rho_i \\ &= \int_{f_i(x)\neq y} \mathbf{1}\{f_i(x) \neq y\} d\rho_i + \int_{f_i(x)=y} \mathbf{1}\{f_i(x) \neq y\} d\rho_i\end{aligned}$$

Here, the first integral is zero since $\rho_i = 0$ when $f_i(x) \neq y$, and the second integral is zero since $\mathbf{1}_{f_i(x)\neq y} = 0$ for the entire domain. Hence $\mathcal{E}_{\rho_i}(f_i) = 0$.

Next, we show $\inf_{f:\mathcal{X}\to\mathcal{Y}} \mathcal{E}_{\rho_i}(f) = 0$. We need to show that no other function f can achieve a lower misclassification excess risk, however since $\mathbf{1}_{f_i(x)\neq y}$ is always zero when $f = f_i$, and ρ_i is also always zero if and only if the prediction was misclassified, all terms are zero. There is no way for a negative excess risk, hence no other function can achieve a lower risk.

Q11 Part (b)

We aim to show:

$$\max_{i=1,\cdots,T} \mathbb{E}_{S\sim\rho_i^n} \mathcal{E}_{\rho_i}(A(S)) \geq \min_{j=1,\cdots,k} \frac{1}{T} \sum_{i=1}^{T} \mathcal{E}_{\rho_i}(A(S_j^i))$$

Taking inspiration from Hint 1, consider the LHS for each i, averaging over all is (using the inequality in Hint 2 twice):

$$\begin{aligned}
\max_{i=1,\cdots,T} \mathbb{E}_{S\sim\rho_i^n} \mathcal{E}_{\rho_i}(A(S)) \geq \frac{1}{T} \sum_{i=1}^{T} \mathbb{E}_{S\sim\rho_i^n} \mathcal{E}_{\rho_i}(A(S)) &= \frac{1}{T} \sum_{i=1}^{T} \mathbb{E}_{S\sim\rho_i^n} \int_{\mathcal{C}\times\mathcal{Y}} \mathbb{1}\{A(S)(x) \neq y\} d\rho_i \\
&= \frac{1}{T} \sum_{i=1}^{T} \int_{\mathcal{C}\times\mathcal{Y}} \mathbb{E}_{S\sim\rho_i^n} \mathbb{1}\{A(S_j^i)(x) \neq y\} d\rho_i \\
&= \frac{1}{T} \sum_{i=1}^{T} \frac{1}{k} \sum_{j=1}^{k} \int_{\mathcal{C}\times\mathcal{Y}} \mathbb{1}\{A(S_j^i)(x) \neq y\} d\rho_i \\
&= \frac{1}{k} \sum_{j=1}^{k} \frac{1}{T} \sum_{i=1}^{T} \mathcal{E}_{\rho_i}(A(S_j^i)) \\
&\geq \min_{j=1,\cdots,k} \frac{1}{T} \sum_{i=1}^{T} \mathcal{E}_{\rho_i}(A(S_j^i))
\end{aligned}$$

Q11 Part (c)

We aim to show:

$$\frac{1}{T} \sum_{i=1}^{T} \mathcal{E}_{\rho_i}(A(S_j^i)) \geq \frac{1}{2} \min_{r=1,\cdots,p} \frac{1}{T} \sum_{i=1}^{T} \mathbb{1}\{A(S_j^i)(v_r) \neq f_i(v_r)\}$$

First we consider the set $S_j' = \{v_1, \cdots, v_p\}$ and its corresponding misclassification risk:

$$\begin{aligned}
\frac{1}{T} \sum_{i=1}^{T} \mathcal{E}_{\rho_i}(A(S_j^i)) &= \frac{1}{T} \sum_{i=1}^{T} \int_{S_j'\times\mathcal{Y}} \mathbb{1}\{A(S_j^i)(v) \neq f_i(v)\} d\rho_i(v, y) \\
&\geq \frac{1}{T} \sum_{i=1}^{T} \frac{1}{p} \sum_{r=1}^{p} \mathbb{1}\{A(S_j^i)(v_r) \neq f_i(v_r)\} d\rho_i(v_r, y) \\
&\geq \frac{1}{2} \min_{r=1,\cdots,p} \frac{1}{T} \sum_{i=1}^{T} \mathbb{1}\{A(S_j^i)(v_r) \neq f_i(v_r)\}
\end{aligned}$$

Q11 Part (d)

We aim to show:

$$\frac{1}{T}\sum_{i=1}^{T}\mathbb{1}\{A(S_j^i)(v_r) \neq f_i(v_r)\} = \frac{1}{2}$$

We start with considering the expression:

$$\mathbb{1}\{A(S_j^i)(v_r) \neq f_i(v_r)\} + \mathbb{1}\{A(S_j^i)(v_r) \neq f_{i'}(v_r)\}$$

Since $x = v_r$ in these expressions, we know $f_i(x) \neq f_{i'}(x)$, and since f can only take 2 values $\{-1, 1\}$, we know exactly one of the terms in the expression will be 1, and the other 0.

Therefore we know:

$$\mathbb{1}\{A(S_j^i)(v_r) \neq f_i(v_r)\} + \mathbb{1}\{A(S_j^i)(v_r) \neq f_{i'}(v_r)\} = 1$$

Now we take the sum from $i = 1, \cdots, T$:

$$\sum_{i=1}^{T}\mathbb{1}\{A(S_j^i)(v_r) \neq f_i(v_r)\} + \sum_{i=1}^{T}\mathbb{1}\{A(S_j^i)(v_r) \neq f_{i'}(v_r)\} = T$$

$$\frac{1}{T}\sum_{i=1}^{T}\mathbb{1}\{A(S_j^i)(v_r) \neq f_i(v_r)\} = \frac{1}{2}$$

3.2 Q11 Part (e)

We let Z be a random variable with values in $[0, 1]$ and $\mathbb{E}[Z] = \mu$. We aim to show that for any $a \in (0, 1)$:

$$\mathbb{P}(Z > 1 - a) \geq \frac{u - (1 - a)}{a}$$

Markov's Inequality states that for any non-negative random variable X and any $a > 0$:

$$\mathbb{P}(X \geq a) \leq \frac{\mathbb{E}[X]}{a}$$

We substitute $X = 1 - Z$, which is permitted as $0 \leq Z \leq 1$ hence X is still non-negative:

$$\begin{aligned}
\mathbb{P}(1 - Z \geq a) &\leq \frac{\mathbb{E}(1 - Z)}{a} \\
\mathbb{P}(1 - a \geq Z) &\leq \frac{1 - \mu}{a} \\
1 - \mathbb{P}(1 - a < Z) &\leq \frac{1 - \mu}{a} \\
\mathbb{P}(Z > 1 - a) &\geq 1 - \frac{1 - \mu}{a} \\
\mathbb{P}(Z > 1 - a) &\geq \frac{\mu - (1 - a)}{a}
\end{aligned}$$

Q11 Part (f)

Let $Z = \mathcal{E}_\rho(A(S))$ and $a = \frac{7}{8}$. This forms the equation:

$$\mathbb{P}_{S\sim\rho^n}\left(\mathcal{E}_\rho(A(S)) > 1 - \frac{7}{8}\right) \geq \frac{\mu - (1-\frac{7}{8})}{\frac{7}{8}}$$
$$\mathbb{P}_{S\sim\rho^n}\left(\mathcal{E}_\rho(A(S)) > \frac{1}{8}\right) \geq \frac{8\cdot\mathbb{E}_{S\sim\rho^n}\mathcal{E}_\rho(A(S)) - 1}{7}$$

Combining parts (b), (c) and (d), we obtain the following inequality:

$$\max_{i=1,\cdots,T}\mathbb{E}_{S\sim\rho_i^n}\mathcal{E}_{\rho_i}(A(S)) \geq \min_{j=1,\cdots,k}\frac{1}{T}\sum_{i=1}^{T}\mathcal{E}_{\rho_i}(A(S_j^i))$$

$$\frac{1}{T}\sum_{i=1}^{T}\mathcal{E}_{\rho_i}(A(S_j^i)) \geq \frac{1}{2}\min_{r=1,\cdots,p}\frac{1}{T}\sum_{i=1}^{T}\mathbb{1}\{A(S_j^i)(v_r) \neq f_i(v_r)\}$$

$$\frac{1}{T}\sum_{i=1}^{T}\mathbb{1}\{A(S_j^i)(v_r) \neq f_i(v_r)\} = \frac{1}{2}$$

Combining (b) and (c), we obtain:

$$\max_{i=1,\cdots,T}\mathbb{E}_{S\sim\rho_i^n}\mathcal{E}_{\rho_i}(A(S)) \geq \min_{j=1,\cdots,k}\frac{1}{2}\min_{r=1,\cdots,p}\frac{1}{T}\sum_{i=1}^{T}\mathbb{1}\{A(S_j^i)(v_r) \neq f_i(v_r)\}$$

Combining on (d), we obtain:

$$\max_{i=1,\cdots,T}\mathbb{E}_{S\sim\rho_i^n}\mathcal{E}_{\rho_i}(A(S)) \geq \min_{j=1,\cdots,k}\frac{1}{2}\min_{r=1,\cdots,p}\frac{1}{2}$$
$$\max_{i=1,\cdots,T}\mathbb{E}_{S\sim\rho_i^n}\mathcal{E}_{\rho_i}(A(S)) \geq \frac{1}{4}$$
$$\frac{1}{7}\left[8\cdot\max_{i=1,\cdots,T}\mathbb{E}_{S\sim\rho_i^n}\mathcal{E}_{\rho_i}(A(S)) - 1\right] \geq \frac{1}{7}$$

We can deduce that there exists a probability distribution ρ such that $\mathbb{E}_{S\sim\rho^n}\mathcal{E}_\rho(A(S)) \geq \mathbb{E}_{S\sim\rho_i^n}\mathcal{E}_{\rho_i}(A(S))$, because we know $\mathcal{E}_{\rho_i}(f_i) = 0 = \inf_{f:\mathcal{X}\to\mathcal{Y}}\mathcal{E}_{\rho_i}(f) = 0$ from part (a).

Therefore, combining all results:

$$\mathbb{P}_{S\sim\rho^n}\left(\mathcal{E}_\rho(A(S)) > \frac{1}{8}\right) \geq \frac{8\cdot\mathbb{E}_{S\sim\rho^n}\mathcal{E}_\rho(A(S)) - 1}{7} \geq \frac{1}{7}\left[8\cdot\max_{i=1,\cdots,T}\mathbb{E}_{S\sim\rho_i^n}\mathcal{E}_{\rho_i}(A(S)) - 1\right] \geq \frac{1}{7}$$

Hence:

$$\mathbb{P}_{S\sim\rho^n}\left(\mathcal{E}_\rho(A(S)) > \frac{1}{8}\right) \geq \frac{1}{7}$$

Q11 Part (g)

Q11g (i)

No-Free-Lunch Theorem Let $\mathcal{X}$ be an input space, and $\mathcal{Y} = \{-1, 1\}$ be the output space. Denote $S = (x_i, y_i)_{i=1}^n$ be a dataset of n input-output pairs.

For any algorithm $A : S \mapsto (f : \mathcal{X} \to \mathcal{Y})$ and any integer $n \in \mathbb{N}$, there exists a distribution ρ over $\mathcal{X} \times \mathcal{Y}$, such that: $\inf_{f:\mathcal{X}\to\mathcal{Y}} \mathcal{E}_\rho(f) = 0$ and $\mathbb{P}_{S\sim\rho^n}\left(\mathcal{E}_\rho(A(S)) > \frac{1}{8}\right) \geq \frac{1}{7}$

The No-Free-Lunch Theorem for machine learning states that there is no universal algorithm which can perform optimally across all possible data sets or distributions. The effectiveness of a machine learning algorithm is strongly dependent on the underlying patterns of the provided data, and requires attention and understanding to select the appropriate algorithm for the particular task.

Q11g (ii)

The No-Free-Lunch theorem implies that the space of all functions $\mathcal{Y}^X$ is not learnable, since this space includes every possible mapping from $\mathcal{X}$ to $\mathcal{Y}$, for any distribution ρ and for any accuracy ϵ or confidence level δ. The No-Free-Lunch theorem states that for any algorithm A, there exists a distribution ρ, where the performance is suboptimal, referring to the accuracy ϵ and tolerance δ.

The Learnability definition implies that there exists an optimal algorithm A (achieving desired ϵ and δ, but tailored to specific function spaces and data distributions. $\mathcal{Y}^X$ is not learnable as it is far too vast for an algorithm to perform desirably well on the space, which is implied by the NFL theorem.

Q11g (iii)

The implications of the No-Free-Lunch theorem with respect to the design of a machine learning algorithm include the need for a context-aware approach to solving problems with machine learning. Whilst there is no universally optimal algorithm, it is important and beneficial to design algorithms by understanding the characteristics of the data and the context in which the problem is presented.