Supervised Learning (COMP0078) - Coursework 1

Group: 17112496 and 17014657

November 22, 2021

1 Part I

1.1 Linear Regression

1. For each of the polynomial bases of dimension k = 1, 2, 3, 4 fit the data set of Figure 1 $\{(1, 3), (2, 2), (3, 0), (4, 5)\}$
   (a) Can be seen in Figure 1

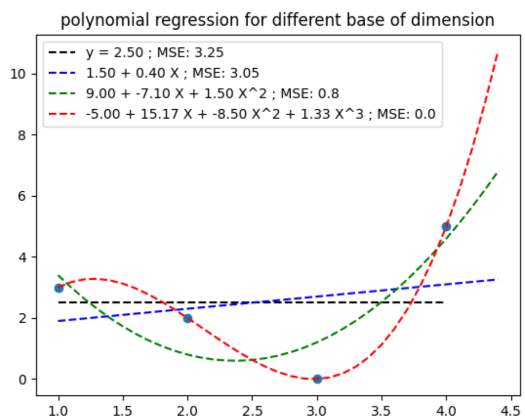

Figure 1: q1a

   (b) $k = 1, y = 2.50$
   $k = 2, y = 1.50 + 0.4x$
   $k = 3, y = 9.00 - 7.10x + 1.50x^2$

   (c) $k = 1, MSE = 3.25$
   $k = 2, MSE = 3.05$
   $k = 3, MSE = 0.8$
   $k = 4, MSE = 0$

2. (a) i. Can be seen in Figure 2

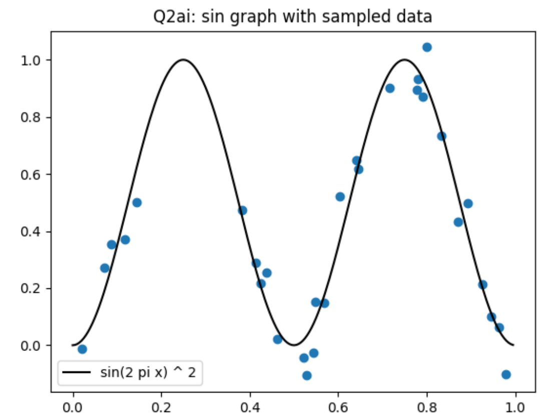

Figure 2: q2ai

ii. Can be seen in Figure 3

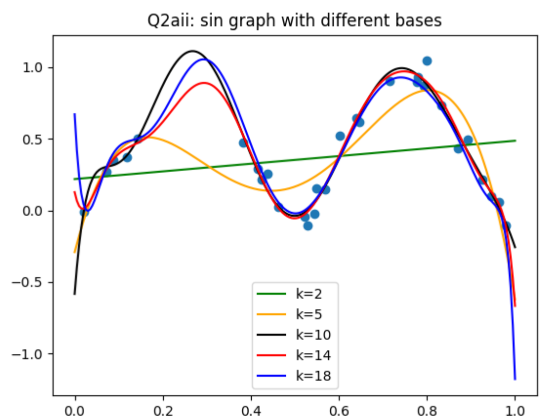

Figure 3: q2aii

(b) Can be seen in Figure 4

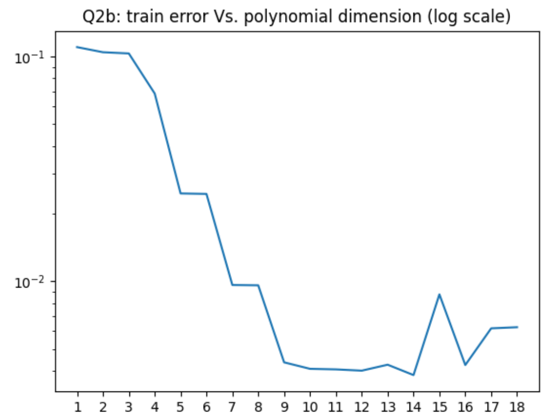

Figure 4: q2b

(c) Can be seen in Figure 5

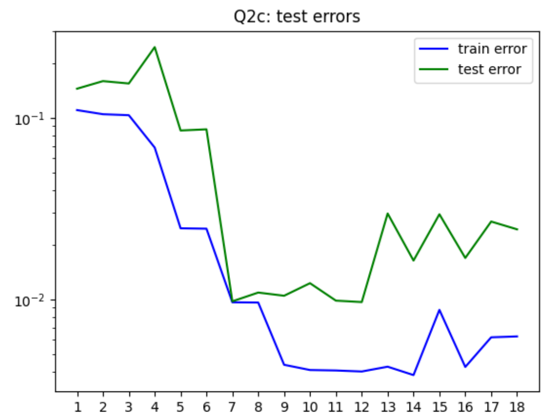

Figure 5: q2c

(d) Can be seen in Figure 6

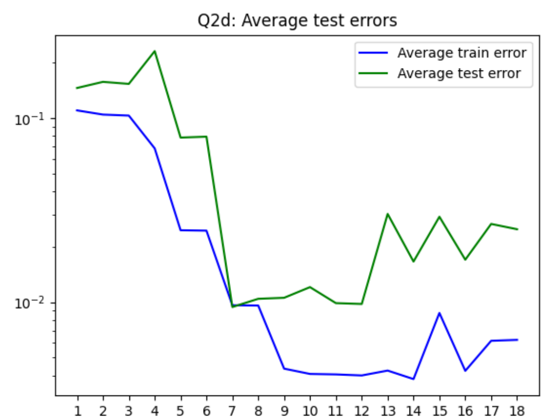

Figure 6: q2d

3. (a) Can be seen in Figure 7

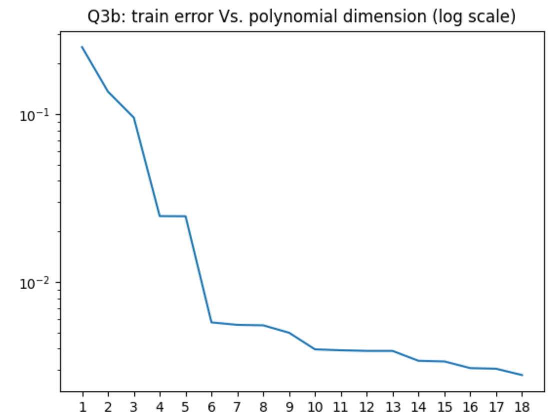

Figure 7: q3b

(b) Can be seen in Figure 8

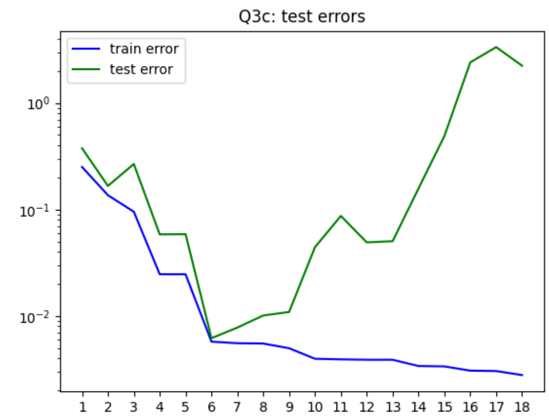

Figure 8: q3c

(c) Can be seen in Figure 9

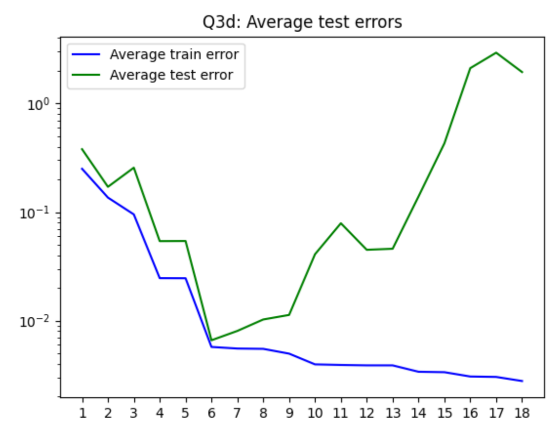

Figure 9: q3d

1.2 Filtered Boston housing and kernels

4. (a) Naive Regression

Train errors (MSE):

[88.53690447203958, 89.05625517409311, 70.78500608716826, 110.1245701557413, 75.36471029588148, 73.20896860824787, 88.53690447203958, 88.53690447203958, 74.69925655385109, 103.65655024393763, 105.4280980079178, 70.78500608716826, 74.69925655385109, 72.82932626320013, 87.83847090333582, 89.17260954648343, 74.26840588336083, 88.44282679075481, 106.0024009595181, 63.04838568298029]
Average Train error: 84.7510408606805

Test errors (MSE):

[96.0692570652203, 78.27586932034362, 133.94430287497312, 37.75248308246793, 111.00481564624556, 107.04731113970813, 96.0692570652203, 96.0692570652203, 123.17101579942707, 48.05021293542669, 51.060640210068954, 133.94430287497312, 123.17101579942707, 109.25364584185289, 80.28144994236538, 77.9115408466843, 124.50934967566639, 79.63233542348257, 46.77672764885909, 152.15572235569434]
Average Test error: 95.30752563066635

(b) The constant function above would be the equation for the constant line of best fit.

(c) Linear Regression With Single Attributes

Attribute 1
Average train error: 70.10134533859306
Average test error: 406.1866450735825

Attribute 2
Average train error: 70.55060644888695
Average test error: 90.07807129548911

Attribute 3
Average train error: 65.99736917028169
Average test error: 70.10536865939437

Attribute 4
Average train error: 74.56165781318867
Average test error: 111.24052869283207

Attribute 5
Average train error: 66.04529347586535
Average test error: 84.61025945589502

Attribute 6
Average train error: 43.774942215028844
Average test error: 52.071679720930305

Attribute 7
Average train error: 72.22384450431682
Average test error: 81.57358107524264

Attribute 8
Average train error: 75.16444574027693
Average test error: 99.41503162245904

Attribute 9
Average train error: 68.92459079506266
Average test error: 102.61772166953749

Attribute 10
Average train error: 64.07736166379607
Average test error: 76.74866727080715

Attribute 11
Average train error: 61.642927454087996
Average test error: 68.85765137708623

Attribute 12
Average train error: 34.63832369177509
Average test error: 52.6028828382758

(d) Linear regression With All Attributes

Train errors (MSE): [9.800912010561877, 26.41591453604629, 28.533435672808128, 17.928741472128173, 26.587391563068284, 28.56298682045523, 21.790414285072067, 14.885261407621142, 9.545884852671335, 27.38234713777917, 23.505212631206412, 22.439537893676352, 23.899065611028476, 14.920959277547137, 24.494070224458454, 18.04472967982342, 10.237562396685968, 11.10043268386182, 14.388762113087475, 18.60683054480178]
Average Train error: 19.65352264071945

Test errors (MSE): [308.4305876368855, 18.26657842371656, 11.725774795025995, 60.81757653068132, 17.59874918401897, 12.086840331093985, 31.602815782219825, 42.326888139556026, 163.6238612714349, 17.816843886636786, 25.897603523904213, 28.565073773786946, 24.728830474274822, 42.28771389679954, 22.962207592402397, 55.960839113493755, 72.63883190213198, 65.80202350229514, 45.86768901730572, 50.343174543957694]
Average Test error: 55.9675251660811

1.3 Kernelised ridge regression

5. (a) Best predictor: [8192, 9.094947017729282e-13, 2.4752228156226206e-08]

   (b) Can be seen in Figure 10

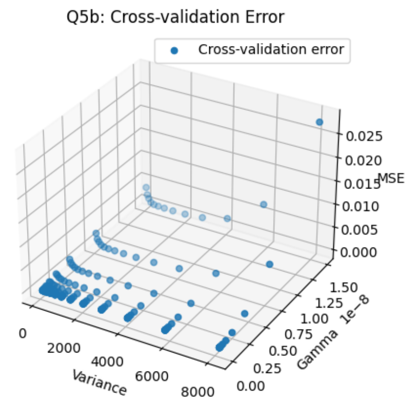

Figure 10: q5b

   (c) MSE on training set with predictor: [8192, 9.094947017729282e-13, 2.4752228156226206e-08] is [2.7006e-08]

   MSE on testing set with predictor: [8192, 9.094947017729282e-13, 2.4752228156226206e-08] is [7.46052434e-08]

   (d)

Method	MSE train	MSE test
Naive Regression	77.68 ± 16.43	112.48 ± 37.85
Linear Regression (attribute 1)	68.07 ± 14.44	269.71 ± 400.28
Linear Regression (attribute 2)	69.23 ± 24.22	93.54 ± 31.94
Linear Regression (attribute 3)	62.59 ± 12.22	78.91 ± 28.37
Linear Regression (attribute 4)	75.21 ± 16.81	110.51 ± 38.04
Linear Regression (attribute 5)	66.20 ± 14.90	85.85 ± 31.35
Linear Regression (attribute 6)	37.44 ± 14.68	68.64 ± 29.84
Linear Regression (attribute 7)	68.61 ± 16.34	91.78 ± 34.25
Linear Regression (attribute 8)	73.85 ± 17.05	104.97 ± 36.94
Linear Regression (attribute 9)	69.94 ± 14.26	117.07 ± 74.66
Linear Regression (attribute 10)	64.69 ± 12.56	77.45 ± 29.81
Linear Regression (attribute 11)	61.19 ± 10.86	71.95 ± 22.97
Linear Regression (attribute 12)	37.04 ± 8.07	45.96 ± 19.82
Linear Regression (all attributes)	18.59 ± 6.60	78.23 ± 91.66
Kernel Ridge Regression	2.57 ± 3.01	9.73 ± 0.00024

2 Part II

6. Can be seen in Figure 11

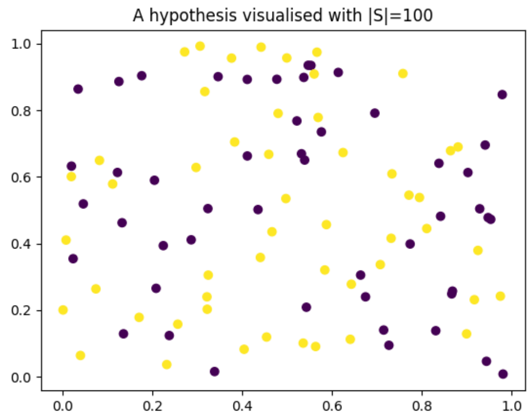

Figure 11: q6

7. (a) Can be seen in Figure 12

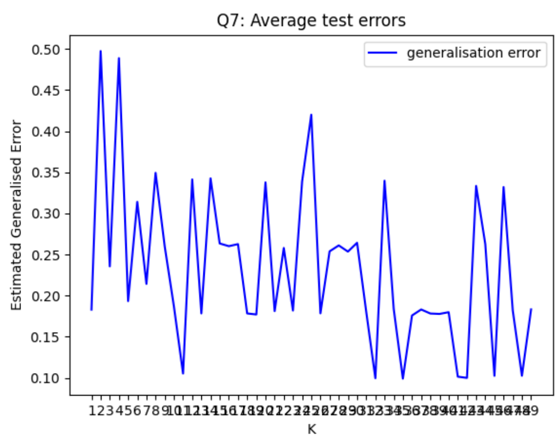

Figure 12: q7

(b) If you look at 12, you can see that the trend oscillates. This is mainly because KNN algorithm performs better (low generalisation error) when K is set to an odd number, compared to when it's set to an even number. Odd number can make the KNN algorithm to calculate much clearer majority when there are two y labels. As K increases, we tend to see lower generalisation error, and this is equivalent to say that the decision boundary gets smoother across different classifications when K is getting bigger.

8. (a) Can be seen in Figure 13

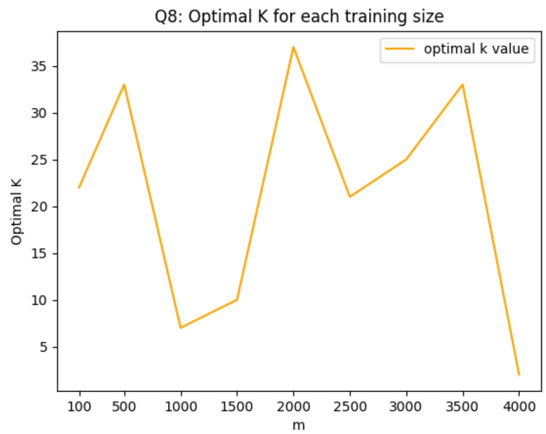

Figure 13: q8

Optimal K value for each m (training size)
m = 100: k = 22,
m = 500: k = 33,
m = 1000: k = 7,
m = 1500: k = 10,
m = 2000: k = 37,
m = 2500: k = 21,
m = 3000: k = 25,
m = 3500: k = 33,
m = 4000: k = 2

(b) From 13 we can see that if the training size is big enough (m=4000), because the decision space is not sparse, optimal k is calculated as a relatively small value (k=2).

9. (a) $K_c(\boldsymbol{x}, \mathbf{z}) := c + \sum_{i=1}^{n} x_i z_i \qquad x, z, \in \mathbb{R}$

For what values of $c \in \mathbb{R}$ is a positive semidefinite (PSD) kernel?

To make a kernel a PSD, it should satisfy the followings:
1. Kernel should be symmetric
2. Matrix $k_c(x, z)$ is PSD

First, let's take a look if this kernel is symmetric.

$K_c(\boldsymbol{x}, \mathbf{z})$
$= x_1 x_1 + x_2 z_2 + \cdots + x_n z_n + c$

This can be written as:

$$\begin{pmatrix} x_1 \\ \vdots \\ x_n \\ \sqrt{c} \end{pmatrix} \cdot \begin{pmatrix} z_1 \\ \vdots \\ z_n \\ \sqrt{c} \end{pmatrix} = \begin{pmatrix} z_1 \\ \vdots \\ z_n \\ \sqrt{c} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_n \\ \sqrt{c} \end{pmatrix} = k_c(x, z)$$

$\therefore$ Kernel is symmetric

$<$ Matrix $k_c(x, z)$ is PSD $>$

$\rightarrow$ given that $k_c(x, z) = \langle \emptyset(x), \emptyset(z) \rangle$

To prove that the kernel function is PSD, we need to show that:
$\sum_{i,j=1}^{m} a_i a_j K(\mathbf{x}_i, \mathbf{x}_j) = \left\langle \sum_{i=1}^{m} a_i \phi(\mathbf{x}_i), \sum_{j=1}^{m} a_j \phi(\mathbf{x}_j) \right\rangle = \|\sum_{i=1}^{m} a_i \phi(\mathbf{x}_i)\|^2 \geq 0$

$\sum_{i,j=1}^{m} a_i a_j \left(\sum_{k=1}^{n} x_{ik} x_{jk} + c\right)$

$= \sum_{i,j=1}^{m} a_i a_j \left(\sum_{k=1}^{n} x_{ik} x_{jk}\right) + \sum_{i,j=1}^{m} a_i a_j \,(c)$

$= \sum_{k}^{n} \sum_{i,j}^{m} a_i x_{ik} a_j x_{jk} + c \left(\sum_{i=1}^{m} a_i\right)^2$

$= \sum_{k}^{n} \sum_{i}^{m} a_i x_{ik} \sum_{j}^{m} a_j x_{jk} + c \left(\sum_{i=1}^{m} a_i\right)^2$

$= \sum_{k}^{n} \left\langle \sum_{i}^{m} a_i x_{ik}, \sum_{j}^{m} a_j x_{jk} \right\rangle + c \left(\sum_{i=1}^{m} a_i\right)^2$

$= \sum_{k}^{n} ||\sum_{i}^{m} a_i x_{ik}||^2 + c \left(\sum_{i=1}^{m} a_i\right)^2 \geq 0$

$\therefore$ The final condition that makes $k_c(x, z)$ PSD is:

$c \geq \frac{-\sum_{k}^{n} \left\|\sum_{i}^{m} a_i x_{ik}\right\|^2}{\left(\sum_{i}^{m} a_i\right)^2}$

This means that $c \geq 0$ makes $k_c(x, z)$ a PSD Kernel

(b) $K_c(\boldsymbol{x}, \mathbf{z}) := c + \sum_{i=1}^{n} x_i z_i \qquad x, z, \in \mathbb{R}$

If we use this kernel function with linear regression (least squares), c will act as a regulariser.

Say,
$k_c(x, z) = c + A(x, z) \qquad$,where $A(x, z) = \sum_{i=1}^{n} x_i z_i$

From equation (11) in the coursework brief, we know that the dual optimisation formulation after kernelization (without regularisation) is

$$\boldsymbol{\beta}^* = \text{argmin}_{\boldsymbol{\beta} \in \Re^{\ell}} \frac{1}{\ell} \sum_{i=1}^{\ell} \left(\sum_{j=1}^{\ell} \alpha_j K_{i,j} - y_i \right)^2$$

$$= argmin \frac{1}{\ell} \sum_{i}^{\ell} \left(\sum_{j}^{\ell} a_j (c + A(x, z)) - y_i \right)^2$$

$$= argmin \frac{1}{\ell} \sum_{i}^{\ell} \left(c \sum_{j}^{\ell} a_j + \sum_{j}^{\ell} a_j A(x, z) - y_i \right)^2$$

Since,
$(A + B + C)^2 = (A + (B + C))^2$

$$= argmin \frac{1}{\ell} \sum_{i}^{\ell} \left(c^2 \left(\sum_{j}^{\ell} a_j \right)^2 + \left(\sum_{j}^{\ell} a_j A(x, z) - y_i \right)^2 + 2c \left(\sum_{j}^{\ell} a_j \right) \left(\sum_{j}^{\ell} a_j A(x, z) - y_i \right) \right)$$

From the last optimisation formulation, we can see that

1.
$\left(\sum_{j}^{\ell} a_j A(x, z) - y_i \right)^2$ is just a Kernalised Linear Regression optimisation without the variable c
You can obtain this formulation when $c = 0$

2.
$c^2 \left(\sum_{j}^{\ell} a_j \right)^2 + 2c \left(\sum_{j}^{\ell} a_j \right) \left(\sum_{j}^{\ell} a_j A(x, z) - y_i \right)$ are the extra terms that affects x

$\therefore$ c is a regularisation term for the optimisation formulation.

10. We were given the following kernel
$K_\beta(\boldsymbol{x}, \mathbf{t}) = \exp\left(-\beta\|\boldsymbol{x} - \mathbf{t}\|^2\right)$

To simulate 1NN in Gaussian kernelised linear regression, we first assume x' and t' is the x input of the 1NN, which can be shown as below.

$\sum_{i=1}^{m} \alpha_i \exp\left(-\beta\|x' - t'\|^2\right) = \sum_{i=1}^{m} \alpha_i \exp\left(-\beta\|x - t\|^2\right)$

11. First, we can think about this problem as a using matrix. We make an nxn matrix and then fill in the entries with 0 if the mole is hiding in the hole or 1 if mole is out of the hole. By doing this, we can have a initial configuration of the matrix.
Let's call this as an Intitial matrix I.

Then we can make another matrix that looks like $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$.
Let's call this as a Hit matrix H.
We will propagate this matrix H to our initial matrix I by turns, and then our final goal is to change the initial matrix to a null matrix, which is a matrix full of zeros.

When propagating the Hit matrix H, we are going to do a element-wise XOR operations.
That way, overloading <. , .> notation for XOR operation, we can do:
<0, 1>=1
<1, 0>=1
<0, 0>=0
<1, 1>=0

So, our final goal is below:
$\mathbf{I} + \sum_{i,j} \mathbf{H}_{ij} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

This algorithm is solvable in polynomial time.