Coursework 1

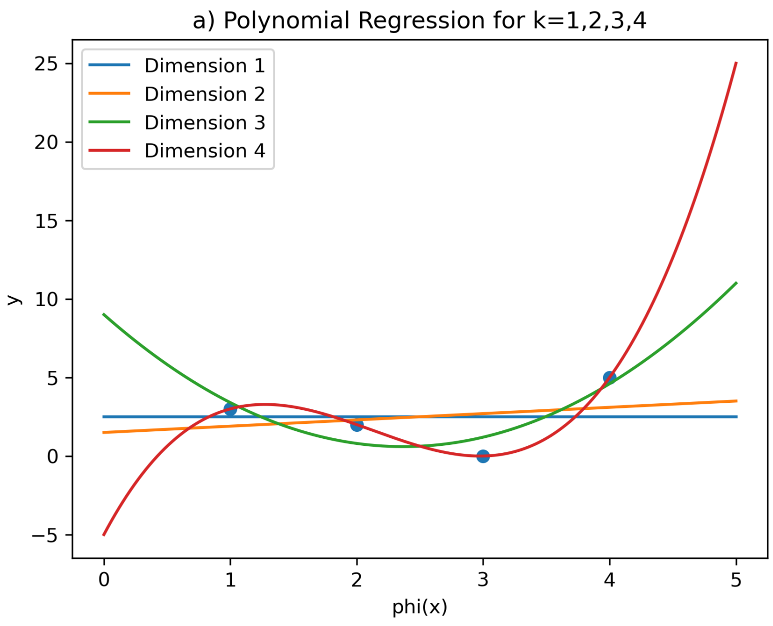
STAT0028: Supervised Learning

SID: 25226382 and 25150451

Due: November 20, 2025

Part I

1. (a)



(b) As in the code, the equations are:

For $k = 1$:

$$y = 2.5$$

For $k = 2$:

$$y = 1.5 + 0.4x$$

For $k = 3$:

$$y = 9.0 - 7.1x + 1.5x^2$$

For $k = 4$ as given:

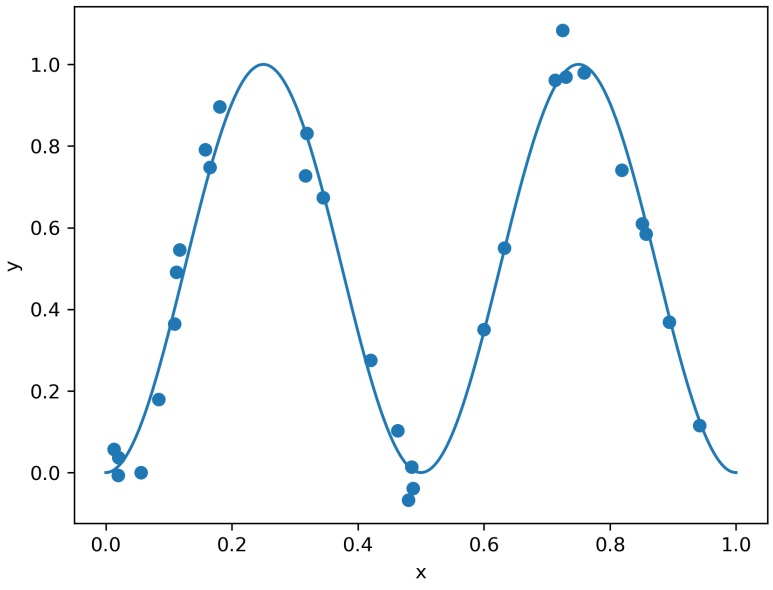
$$y = -5.0 + 15.17x - 8.5x^2 + 1.33x^3$$

(c)

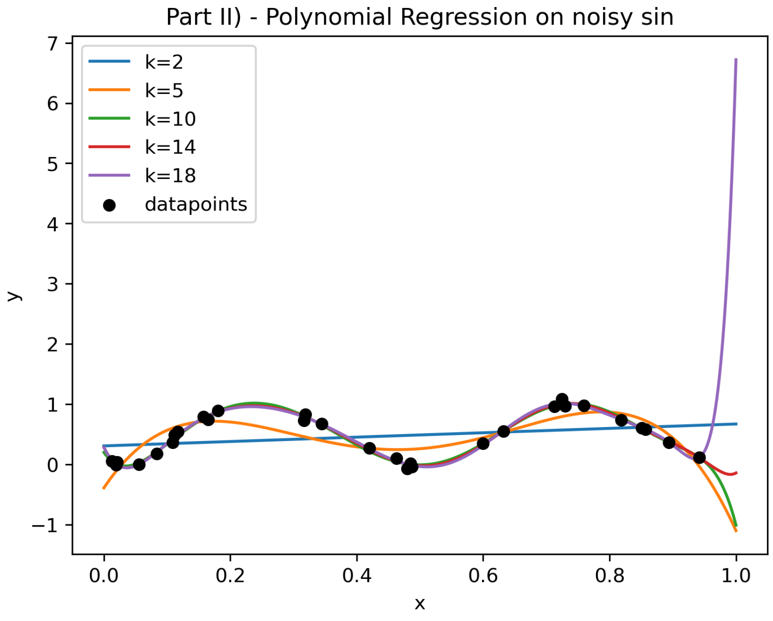
Mean Squared Errors by k:

- $k = 1$: MSE $= 3.25$
- $k = 2$: MSE $= 3.05$
- $k = 3$: MSE $= 0.80$
- $k = 4$: MSE $= 0.00$

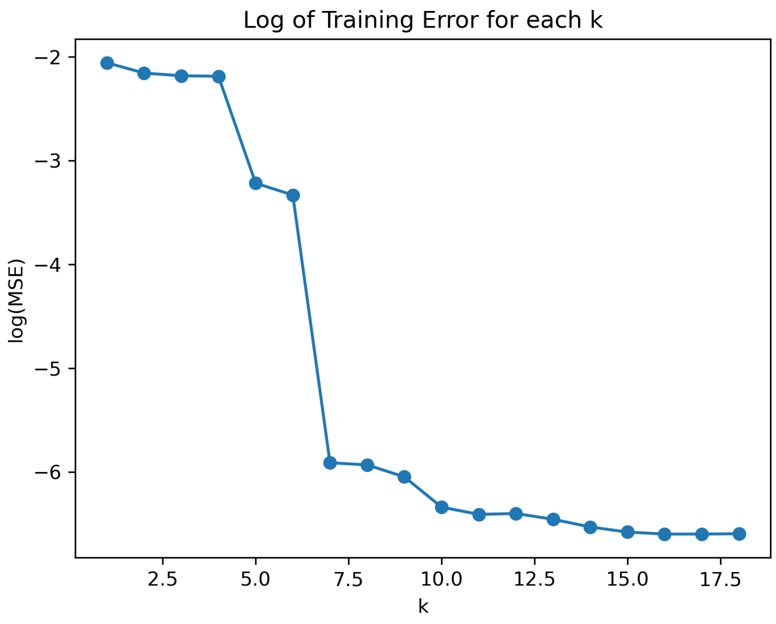
2. (a) i)



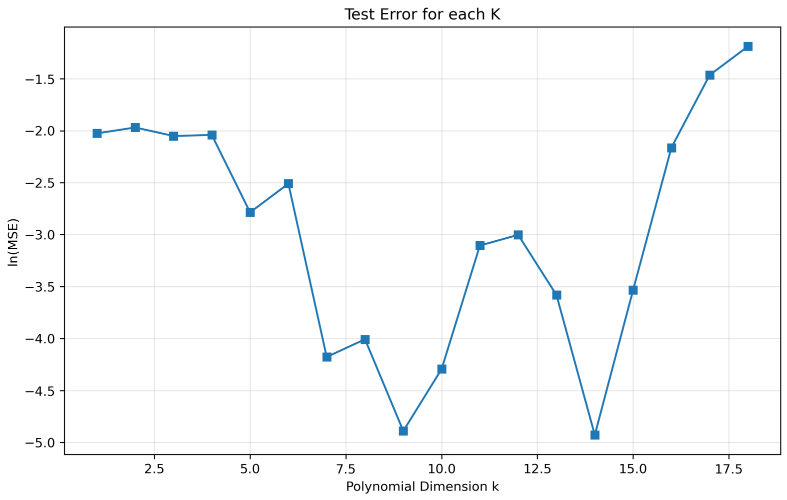
ii)



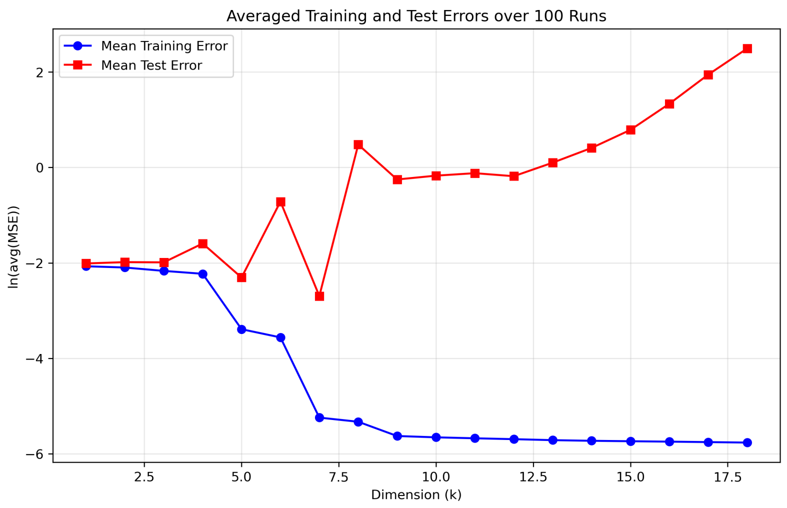
(b)



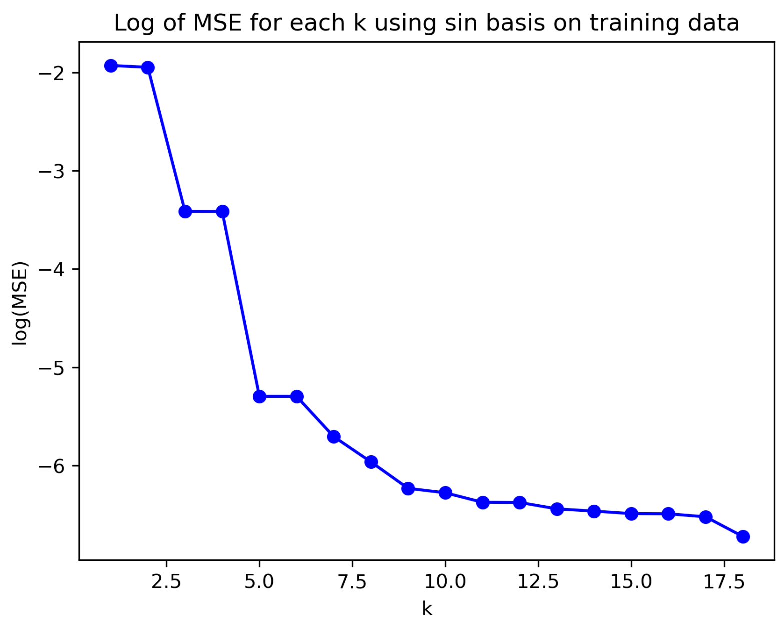
(c)



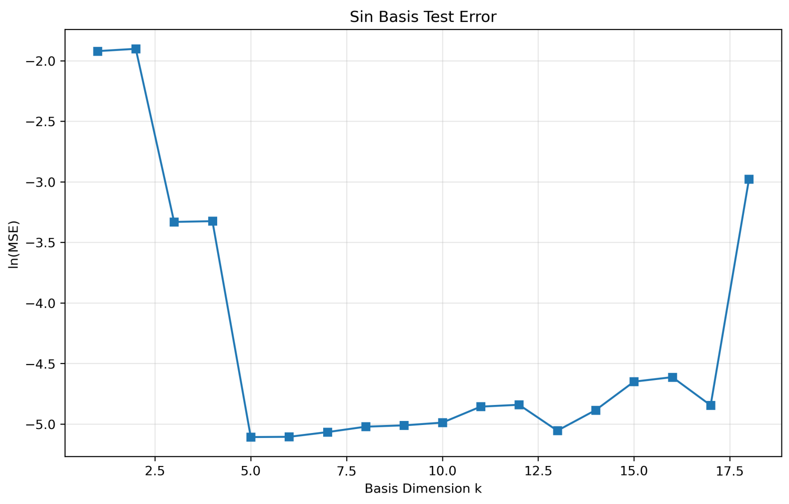
(d) We present the training and test errors on the same plot replicating part b and c over 100 runs:



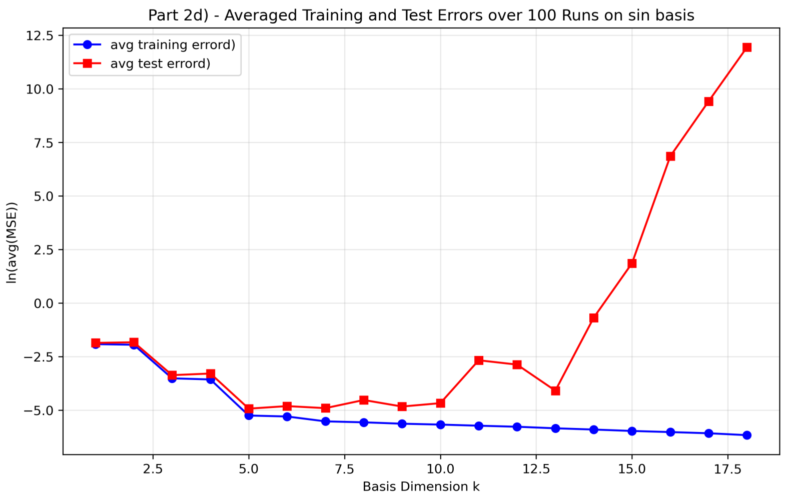
3. Repeated 2b)



Repeated 2c)



Repeated 2d)



4. (a) Average MSE on training data: 84.543, Average MSE on test data: 84.4704

   (b) The constant function predicts the mean of the target variable across all observations, it predicts the same value (the mean) for any given x.

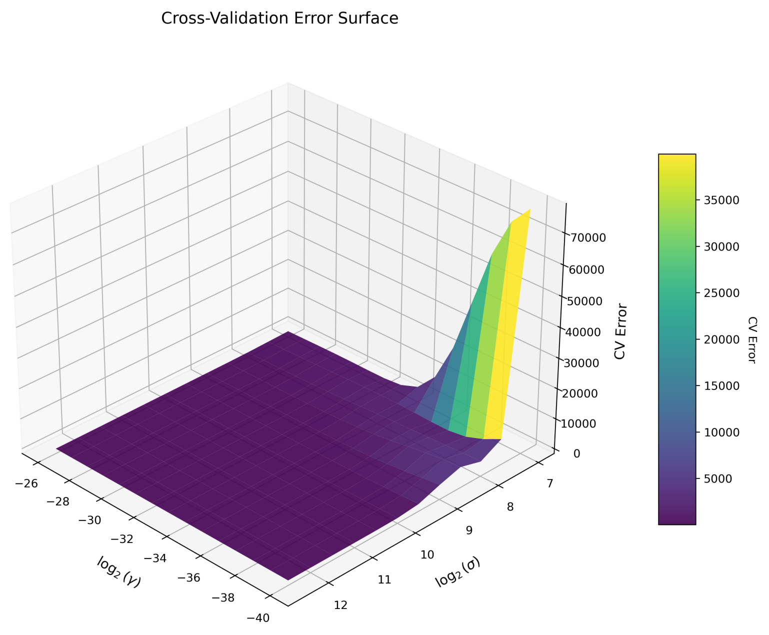
   (c)

Attribute	**Train MSE**	**Test MSE**
CRIM	71.2493	73.9121
ZN	74.1934	72.3831
INDUS	64.9155	64.5597
CHAS	82.0169	82.1931
NOX	69.4759	68.4828
RM	43.3571	44.4611
AGE	72.6852	72.2589
DIS	79.4388	78.9768
RAD	71.9285	73.0795
TAX	65.5870	66.9681
PTRATIO	62.3501	63.8327
LSTAT	38.6980	38.5040

   (d) Average MSE on training data: 22.2799, Average MSE on test data: 23.9752

5. (a) Best gamma: $1.4901161193847656 \times 10^{-8}$, Best sigma: 1448.1546878700494

(b)



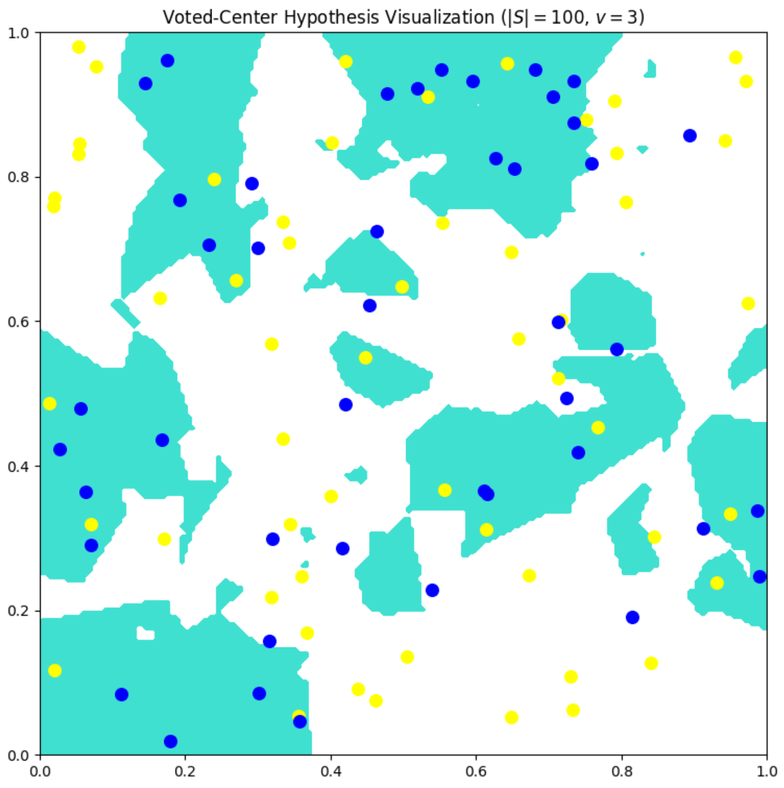
(c) MSE using aforementioned gamma and sigma: Training set = 8.8579, Test set = 11.2459

(d)

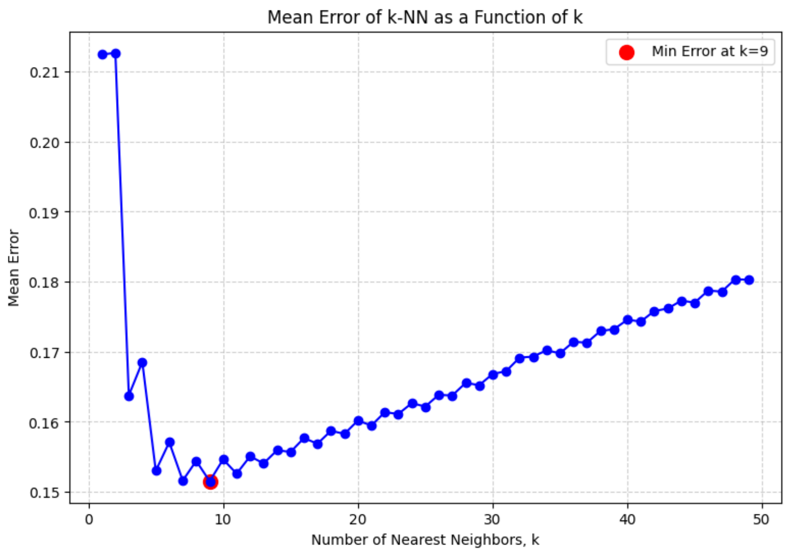
Method	**MSE Train**	**MSE Test**
Naive Regression	84.5430 ± 5.3890	84.4704 ± 10.7556
Linear Regression (attribute 1)	71.2493 ± 4.8944	73.9121 ± 10.4003
Linear Regression (attribute 2)	74.1934 ± 4.3434	72.3831 ± 8.7708
Linear Regression (attribute 3)	64.9155 ± 4.3465	64.5597 ± 8.8503
Linear Regression (attribute 4)	82.0169 ± 5.1223	82.1931 ± 10.6225
Linear Regression (attribute 5)	69.4759 ± 4.4451	68.4828 ± 8.9766
Linear Regression (attribute 6)	43.3571 ± 3.0174	44.4611 ± 5.9058
Linear Regression (attribute 7)	72.6852 ± 4.8133	72.2589 ± 9.6762
Linear Regression (attribute 8)	79.4388 ± 5.2619	78.9768 ± 10.5443
Linear Regression (attribute 8)	71.9285 ± 4.8839	73.0795 ± 9.8510
Linear Regression (attribute 9)	65.5870 ± 4.4846	66.9681 ± 9.0169
Linear Regression (attribute 10)	62.3501 ± 3.9896	63.8327 ± 8.0896
Linear Regression (attribute 11)	38.6980 ± 2.3338	38.5040 ± 4.6313
Linear Regression (all attributes)	22.2799 ± 1.6495	23.9752 ± 3.5857
Kernel Ridge Regression	8.3772 ± 1.9738	12.5495 ± 1.5783

Part II

6.



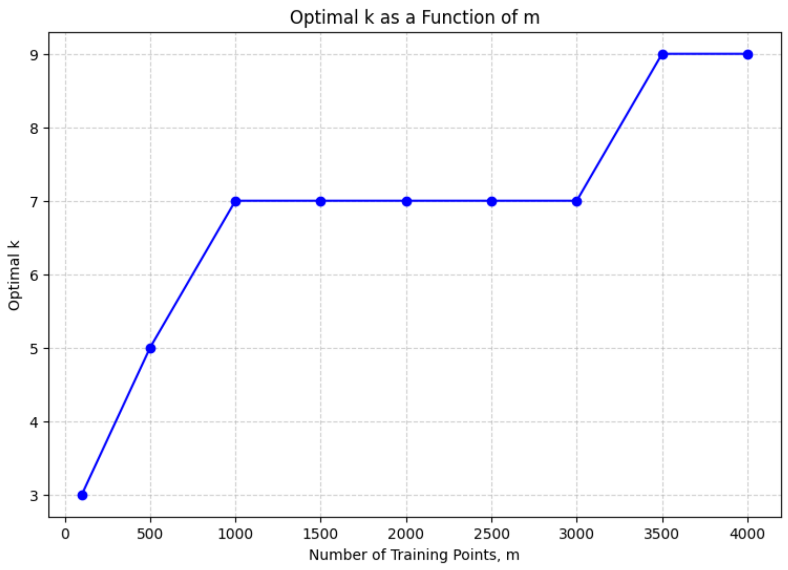
7.



The curve shows a noisy U-shape with a minimum at $k = 9$. This depicts the Bias-Variance Trade-off in k-NN as, for fixed m, increasing k both reduces variance and increases bias. By increasing k, more samples are used for prediction and so any noise on any single sample has less impact on the overall predictive performance, thereby reducing variance. However, by using more samples for prediction (with a fixed m), a larger more generalised neighbourhood is used for prediction, which can smooth out local structure and mask the complexity of the true underlying function, which increases bias.

The noisiness of the U-shape is due to the random selection of classification labels in tie cases for kNN with even k. Here, the curve shows a minimum at just over 0.15, where an optimal algorithm could reach an error of 0.1 as in expectation given 20% of the samples are randomly sampled from 0,1, we would expect half of those to still be correctly labelled by chance - leaving 10% minimum error.

8.



The curve shows a monotonically increasing shape - as m (the number of training points) increases, so does the optimal k. This is because increasing m means increasing the number of samples in the local neighbourhood for prediction, and so more samples can be used without taking samples from further away from the inference point, which would overly smoothing out local structure. In other words, increasing m allows a larger increase in k before bias dominates the bias-variance tradeoff.

Part III

9. (a) Consider the following function,

$$\begin{aligned} K_1(x, z) &= \sum_{i=1}^{n} x_i z_i \\ &= \langle x, z \rangle \end{aligned}$$

$K_1(x, z) = \langle x, z \rangle$ is of the form $K(x, t) = \langle \phi(x), \phi(t) \rangle$, and so is a positive semi definite (PSD) kernel.

Now consider,

$$\begin{aligned} K_2(x, z) &= c \\ &= \langle \sqrt{c}, \sqrt{c} \rangle \end{aligned}$$

$K_2(x, z) = \langle \sqrt{c}, \sqrt{c} \rangle$ is of the form $K(x, t) = \langle \phi(x), \phi(t) \rangle$ so long as $c \geq 0$, as $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$. Therefore, $K_2(x, z) = c$ is also a PSD kernel for $c \geq 0$.

Finally, a kernel $K = K_1 + K_2$, where K_1 and K_2 are PSD, is also itself PSD.

Therefore, we can show that $K_c(x, z)$ is a PSD kernel for $c \geq 0$:

$$\begin{aligned} K &= K_1(x, z) + K_2(x, z) \\ &= c + \sum_{i=1}^{n} x_i z_i \\ &= K_c(x, z) \end{aligned}$$

(b) We can rewrite $K_c(x, z)$ as follows:

$$\begin{aligned} K_c(x, z) &= c + \sum_{i=1}^{n} x_i z_i \\ &= c + \sum_{i=1}^{n} x_i z_i \\ &= \begin{bmatrix} \sqrt{c} \\ x \end{bmatrix}^T \begin{bmatrix} \sqrt{c} \\ z \end{bmatrix} \\ &= \langle \phi(x), \phi(z) \rangle \end{aligned}$$

Then we can perform linear regression using the kernel function K_c by using the mapped data $\phi(x)$ as follows:

$$\begin{aligned} f(x) = w^T \phi(x) &= w^T \begin{bmatrix} \sqrt{c} \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \\ &= w_0\sqrt{c} + w_1 x_1 + \cdots + w_n x_n \end{aligned}$$

Here the term $w_0\sqrt{c}$ is the intercept, and so for $c = 0$ this is forced through the origin, whereas for $c \neq 0$ the intercept can be learned from the data.

10. The regression classifier can be written as follows:

$$C_{reg} = sign(f(t)) = sign(\sum_{i=1}^{n} \alpha_i K_\beta(x_i, t))$$

Where $K_\beta(x_i, t) = e^{-\beta \|\mathbf{x}-\mathbf{t}\|^2}$ and $\beta = \hat{\beta}(\mathbf{x_1}, \ldots, \mathbf{x_m}, \mathbf{t})$.

A 1-nearest neighbour classifier can be written as follows:

$$C_{1NN} = y(x^*), \quad x^* = \underset{m}{argmin} \|x_m - t\|^2, \quad y \in \{-1, 1\}$$

When $\alpha_{i=NN} K_\beta(x_{i=NN}, t) >> \alpha_{i\neq NN} K_\beta(x_{i\neq NN}, t)$, it is clear that $sign(f(t))$ collapses to $sign(\alpha_{i=NN} K_\beta(x_{i=NN}, t))$ which gives the correct classification when the sign of α_{NN} is the same as the nearest neighbour classification (as $K_\beta \geq 0$ for all β).

Since $\|x_{i=NN} - t\|^2 < \|x_{i\neq NN} - t\|^2$, as $\beta \to \infty$:

$$\begin{aligned}
\lim_{\beta\to\infty} \frac{K_\beta(x_{i\neq NN}, t)}{K_\beta(x_{i=NN}, t)} &= \lim_{\beta\to\infty} \frac{e^{-\beta\|\mathbf{x_{i\neq NN}}-\mathbf{t}\|^2}}{e^{-\beta\|\mathbf{x_{i=NN}}-\mathbf{t}\|^2}} \\
&= \lim_{\beta\to\infty} e^{-\beta(\|\mathbf{x_{i\neq NN}}-\mathbf{t}\|^2 - \|\mathbf{x_{i=NN}}-\mathbf{t}\|^2)} \\
&= 0 \\
\therefore \quad C_{reg} &\approx sign(\alpha_{i=NN} K_\beta(x_{i=NN}, t))
\end{aligned}$$

Note that the scale of the α terms will not affect the nearest neighbour selection mechanism produced by the kernel K_β. During training, the coefficients α_i are determined by minimising the squared error. For every training point x_j, as $\beta \to \infty$, the kernel centered at the same training point $K_\beta(x_j, x_j) = 1$ while all other kernels $K_\beta(x_j, x_{i\neq j}) \to 0$. Therefore, $f(x_j) \approx \alpha_j K_\beta(x_j, x_j) \approx \alpha_j$ and so $\alpha_j \approx y_j \in \{-1, 1\}$.

Therefore for $\beta \to \infty$, C_{reg} can be trained to simulate C_{1NN}.

11. The generalised Whack-A-Mole game is defined mathematically below:

$$\begin{aligned}
g &= \mathbb{R}^{n\times n} \quad \text{represents a n by n grid} \\
g_{i,j} &= \begin{cases} 1 & \text{if a mole is present at the hole at grid element i,j} \\ 0 & \text{if a mole is not present at the hole at grid element i,j} \end{cases} \\
s &= \mathbb{R}^{n\times n} \quad \text{represents a set of strike locations} \\
s_{i,j} &= \begin{cases} 1 & \text{if the hole at grid element i,j is struck an odd number of times} \\ 0 & \text{if the hole at grid element i,j is not struck, or struck an even number of times} \end{cases}
\end{aligned}$$

If a hole is struck, it and its adjacent holes should flip their state of there being a mole present or not. We want to find the set of strike locations s that results in a final grid $g_f = 0^{n\times n}$ (an $n \times n$ matrix with 0 in every entry, i.e. no moles present) given an arbitrary initial grid g_0.

Note that two strikes in the same spot should cancel each other out Also note that the order of the strikes does not matter, as the effect of state flips are commutative. For these reasons, we choose to work in mod 2 space.

To solve this problem, we construct the following system of equations:

$$
\begin{aligned}
G_f &= 0 = G_0 \oplus AS \\
G_i &= vec(g_i) \\
S &= vec(s) \\
A &= \mathbb{R}^{N \times N} \quad \text{represents the augmented adjacency matrix of G} \\
N &= n^2 \\
A_{k,l} &= \begin{cases} 1 & \text{if the hole at } i = k \,\%\, n, j = k \,//\, n \text{ is at or adjacent to the hole at } i = l \,\%\, n, j = l \,//\, n \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

($\oplus$ represents addition mod 2 (i.e. XOR), % represents the modulo operator, and // the quotient operator)

The system of equations is solved by rearranging to form:

$$
\begin{aligned}
0 &= G_0 \oplus AS \\
AS &= G_0 \\
S &= A^{-1} G_0
\end{aligned}
$$

Note that when rearranging in the first step shown above, we drop the $-$ from G_0. As we are working in mod 2 space, $G_0 = -G_0$.

This can be solved in $O(N^3) = O(n^6)$ time, as the limiting operation is the inversion of the adjacency matrix, which can be done in $O(m^3)$ time via Gaussian Elimination, where m is the number of rows (or columns) of a square matrix.

The matrix s can be recovered by reshaping our solution S, with $s = vec^{-1}(S)$. If a solution does not exist as a result of Gaussian Elimination, then there is no s that solves our problem.