Assignment 1

Callum Lau, 19102521 & Charita Dellaporta, 19025680
Supervised Learning

November 14, 2019

Problem 1.

Solution Part (a)

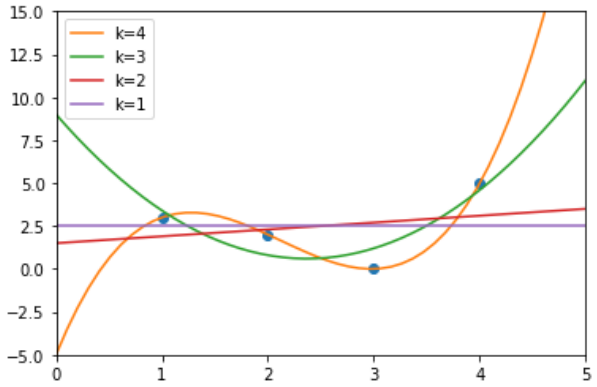

Figure 1: Plot of polynomial fits for k = 1, 2, 3, 4

Part (b) & (c)

Polynomial fit equations and MSE		
k value	Equation	MSE
1	2.5	3.25
2	$1.50 + 0.4x$	3.05
3	$9.0 - 7.1x + 1.5x^2$	0.8
4	$-5 + 15.17x - 8.50x^2 1.33x^3$	0.0

Problem 2.

Solution Part (a) plots for (i) and (ii)

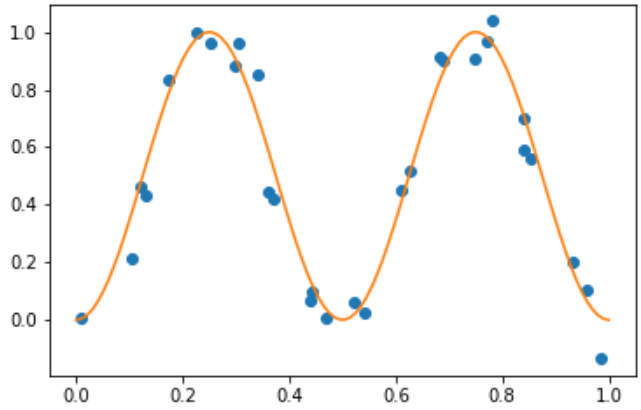

Figure 2: Plot of function $sin^2(2\pi x) + \epsilon$

The plot below shows the 5 polynomial fits to the training points. For large k, i.e. k=14 and k=18 we can see already the phenomenon of overfitting. The curves become increasingly 'erratic' in order to fit closely to all the training points. This suggests that for new test points, we already know that the test errors are going to increase substantially.

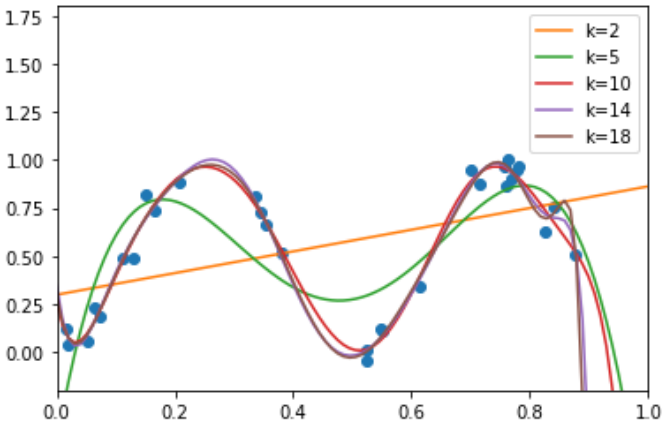

Figure 3: Plot of polynomial fits for k = 2, 5, 10, 14, 18

Part (b) As shown, the training error is a monotonically decreasing function of the polynomial dimension k.

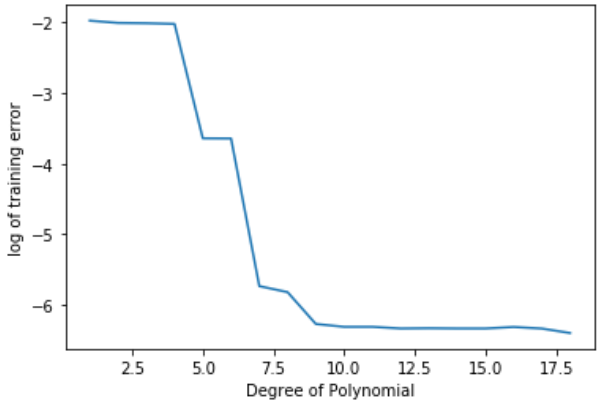

Figure 4: Log of training error vs polynomial dimension k

Part (c) This plot shows in some sense the optimal choice of k to minimise the test error (for one run). For small k, the test error is quite large as the the polynomial gives a very approximate fit to the training data. For larger k, circa k = 6 to 13 the test error decreases (indicating perhaps a suitable polynomial fit to the data). For very large k (k > 13), the test error increases rapidly again as we begin to overfit to the training data.

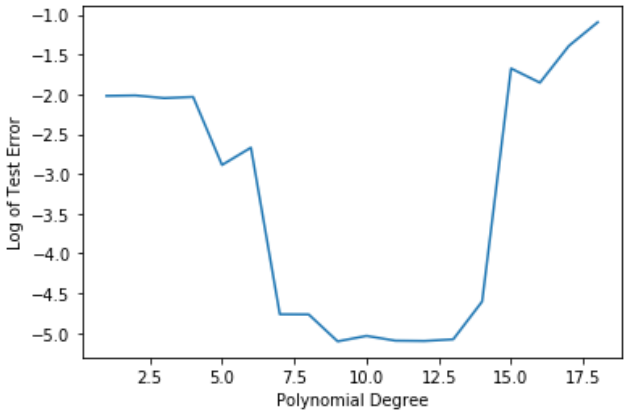

Figure 5: Log of test error vs polynomial dimension k

Part (d) This plot shows the average training and test error over 100 runs for k = 1 to 18. The averages over 100 runs show more clearly the trend of the test error decreasing, before increasing for k > 13.

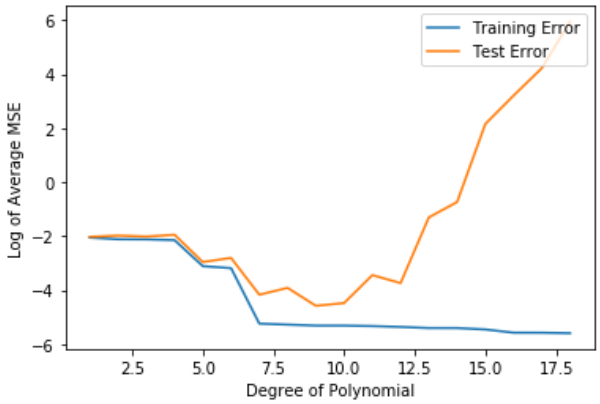

Figure 6: Log(average train/test error) vs polynomial dimension

Problem 3.

Solution Part (b) This is the plot of training error vs basis dimension using the sinusoidal basis $sin(1\pi x), sin(2\pi x), sin(3\pi x), ..., sin(k\pi x)$. As in 2 (b), the training error decreases as the number of basis functions increases.

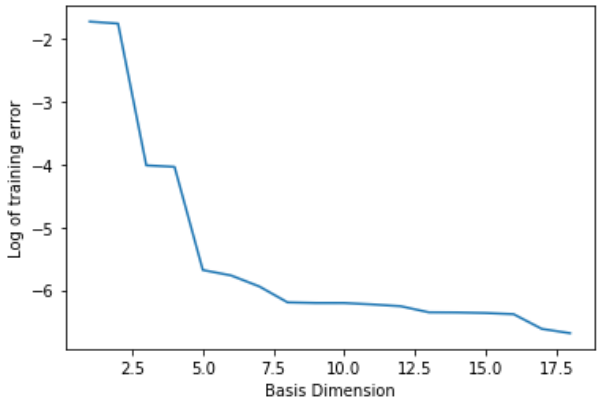

Figure 7: Log of training error vs basis dimension

Part (c) This is the plot of test error vs basis dimension but for one run through the test points. The story is similar to 2 (c), the test error decreasing as the number of basis functions used increases, but eventually for k $>$ 15, the test error begins to increase substantially.

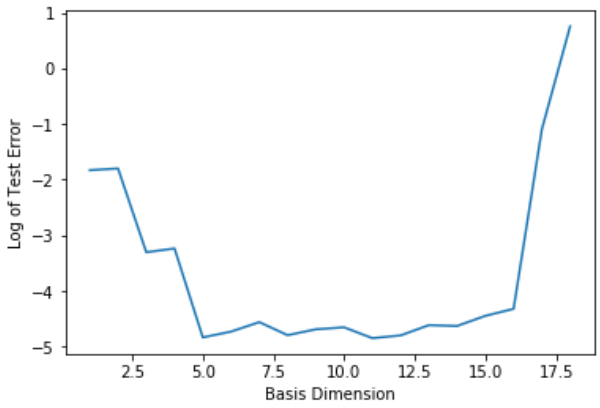

Figure 8: Log of test error vs basis dimension

Part (d) This plot is the average test error over 100 runs, and again shows the clear trend of test error decreasing, before increasing again for too large k.

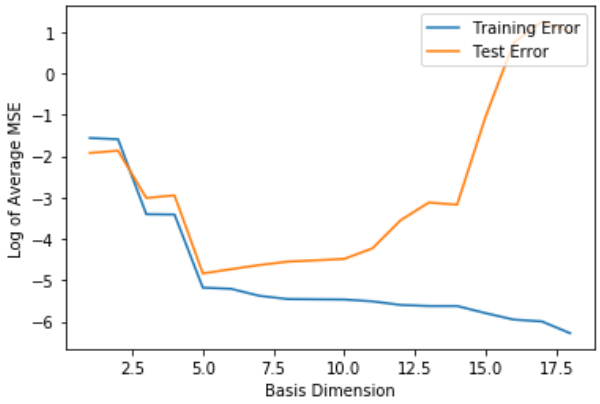

Figure 9: Average(log of training/test error) vs basis dimension

Problem 4.

Solution Part (a)
Over 20 runs, the average naive regression training and test MSE were: Training MSE = 83.6, Test MSE = 86.3

Part (b)
This constant term w is simply the estimate (sampled from training data) of the mean of the median house prices.

Part (c)
This table shows the average of the weight vectors w for each attribute over 20 runs (including the ethically suspect column). The average training and test MSE and their standard deviations for each regression performed with a single attribute are given in Solution 5 (d).

Array of average coefficients of w for each attribute with bias term		
Attribute	w_0	w_1
CRIM	25.35	-0.39
ZN	22.10	0.15
INDUS	31.33	-0.62
CHAS	23.31	6.91
NOX	43.14	-3.16
RM	-32.76	9.54
AGE	32.41	-0.11
DIS	19.51	1.12
RAD	27.75	-0.38
TAX	34.80	-0.02
PTRATIO	65.37	-2.04
B	11.04	0.04
LSTAT	36.54	-0.92

Part (d) Over 20 runs, the average full regression training and test MSE were:
Training MSE = 21.13, Test MSE = 24.98. Clearly incorporating all the attributes will perform better than the single regression in Part (b).

Problem 5.

Solution Part (a) The best parameter values found using 5-fold cross validation over the training data were $\gamma = 2^{-40}$ and $\sigma = 4096$. Note that the best parameters did vary between a few different values for each run because a few combinations gave very similar results. This is made clearer by the 3D scatter plot for part (b).

Part (b) This 3D scatter plot shows all of the average Cross-Validation error for all gamma and sigma pairings. Notably, for both smallest gamma and sigma values the error is extremely large, however for larger gamma and sigma, the error is roughly similar for all values. The second heatmap plot gives a clearer description of how MSE varies with respect to sigma and gamma.

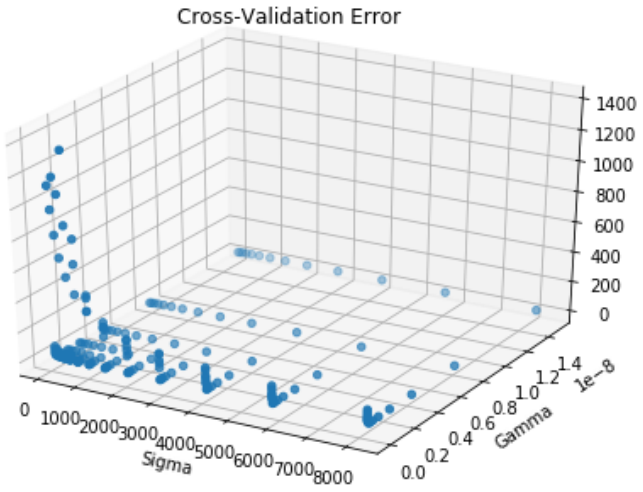

Figure 10: Sigma and gamma value vs average MSE over 5-fold cross-validation

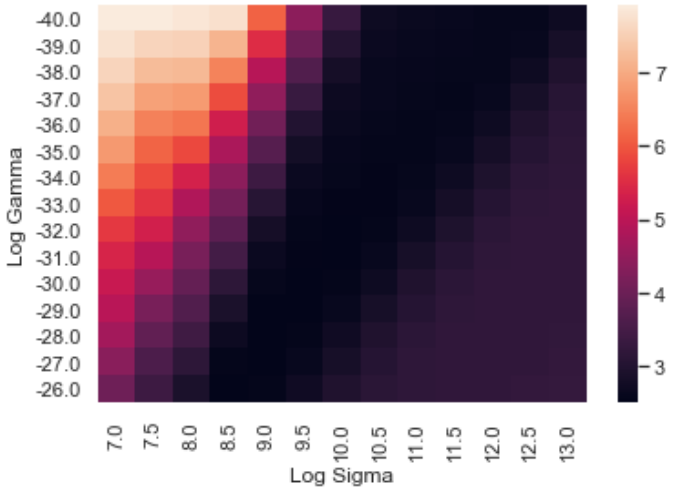

Figure 11: Log base 2 of Sigma and gamma value vs log of average MSE

Part (c) MSE on training and test sets for best γ, σ:
Training MSE = 8.24, Test MSE = 11.95

Part (d) The table of training/test errors is given on the following page.

Table of Regression types with associated training and test errors		
METHOD	**MSE train**	**MSE test**
Naive Regression	83.56 ± 4.18	86.25 ± 8.35
Linear Regression (CRIM)	72.24 ± 3.21	71.17 ± 6.40
Linear Regression (ZN)	74.38 ± 3.54	71.95 ± 6.98
Linear Regression (INDUS)	65.39 ± 3.83	63.53 ± 7.61
Linear Regression (CHAS)	82.16 ± 3.49	81.76 ± 7.22
Linear Regression (NOX)	70.24 ± 3.46	66.90 ± 6.89
Linear Regression (RM)	44.10 ± 3.40	43.07 ± 6.00
Linear Regression (AGE)	73.53 ± 3.87	70.63 ± 7.73
Linear Regression (DIS)	79.97 ± 3.54	77.86 ± 6.95
Linear Regression (RAD)	73.22 ± 3.51	70.24 ± 6.97
Linear Regression (TAX)	66.54 ± 3.49	64.90 ± 6.95
Linear Regression (PTRATIO)	63.57 ± 3.01	61.12 ± 6.05
Linear Regression (B)	75.56 ± 3.67	74.10 ± 7.29
Linear Regression (LSTAT)	38.52 ± 1.97	38.69 ± 4.06
Linear Regression (all attributes)	21.13 ± 1.80	24.98 ± 4.20
Kernel Ridge Regression	8.24 ± 0.67	11.95 ± 1.64

PART II

1 Question 6

1.1 Part a

We have that

$$L_{\boldsymbol{c}}(y, \hat{y}) = [y \neq \hat{y}]_{c_y}$$

so for a set of data points $(x, y) \in (X, [k])$ the expected loss function $\epsilon(f)$ is equal to

$$\begin{aligned} \epsilon(f) = \mathbb{E}(L(f(x), y)) = \\ \sum_{x \in X} \sum_{y \in [k]} [y \neq f(x)] c_y P(x, y) = \\ \sum_{x \in X} \sum_{y \in [k]} [y \neq f(x)] c_y P(y|x) P(x) = \\ \sum_{x \in X} (\sum_{y \in [k]} [y \neq f(x)] c_y P(y|x)) P(x) \end{aligned}$$

Hence, for each particular $\tilde{x} \in X$, we have:

$$\epsilon(f(\tilde{x})) = (\sum_{y \in [k]} [y \neq f(\tilde{x})] c_y P(y|\tilde{x})) P(\tilde{x}) \Rightarrow \epsilon(f(\tilde{x})) \propto \sum_{y \in [k]} [y \neq f(\tilde{x})] c_y P(y|\tilde{x}).$$

Let $\tilde{y} = f(\tilde{x}) \in [k]$ then

$$\begin{aligned} \epsilon(\tilde{y}) \propto \sum_{y \in [k]} [y \neq \tilde{y}] c_y P(y|\tilde{x}) = \\ \sum_{y \in [k] \setminus \{\tilde{y}\}} c_y P(y|\tilde{x}) = \\ \sum_{y \in [k]} c_y P(y|\tilde{x}) - c_{\tilde{y}} P(\tilde{y}|\tilde{x}) \end{aligned}$$

which is minimised (with respect to $\tilde{y}$) when

$$\tilde{y} = \arg\max_{y \in [k]} c_y P(y|\tilde{x})$$

Therefore, for every $x \in X$ the bayes estimator is

$$f^*(x) = \arg\max_{y \in [k]} c_y P(y|x)$$

1.2 Part b

We have

$$L(y, \hat{y}) = |y - \hat{y}|, Y \subset \mathbb{R}$$

so the expected error is equal to

$$\begin{aligned}
\epsilon(f) = \mathbb{E}(L(f(x), y)) = \\
\sum_{x \in X} \sum_{y \in Y} |y - f(x)| P(x, y) = \\
\sum_{x \in X} \sum_{y \in Y} |y - f(x)| P(y|x) P(x) = \\
\sum_{x \in X} (\sum_{y \in Y} |y - f(x)| P(y|x)) P(x).
\end{aligned}$$

So at each point $\tilde{x} \in X$ we have

$$\begin{aligned}
\epsilon(f(\tilde{x})) = \sum_{y \in Y} |y - f(\tilde{x})| P(y|\tilde{x}) P(\tilde{x}) \propto \\
\sum_{y \in Y} |y - f(\tilde{x})| P(y|\tilde{x})
\end{aligned}$$

and by letting $f(\tilde{x}) = \tilde{y} \in Y \subset \mathbb{R}$ we obtain

$$\begin{aligned}
\epsilon(f(\tilde{x})) \propto \sum_{y \in Y} |y - \tilde{y}| P(y|\tilde{x}) = \\
\sum_{y=-\infty}^{\tilde{y}} (\tilde{y} - y) P(y|\tilde{x}) + \sum_{y=\tilde{y}}^{\infty} (y - \tilde{y}) P(y|\tilde{x}).
\end{aligned}$$

Differentiating $\epsilon(f(\tilde{x}))$ with respect to $\tilde{y}$ and setting equal to zero we get:

$$\begin{aligned}
\frac{\partial}{\partial \tilde{y}} \epsilon(f(\tilde{x})) = \sum_{y=-\infty}^{\tilde{y}} P(y|\tilde{x}) + \sum_{y=\tilde{y}}^{\infty} -P(y|\tilde{x}) = 0 \Rightarrow \\
\sum_{y=-\infty}^{\tilde{y}} P(y|\tilde{x}) = \sum_{y=\tilde{y}}^{\infty} P(y|\tilde{x}).
\end{aligned}$$

We know that $P(y|\tilde{x})$ is a probability mass function, so $\sum_{y \in Y} P(y|\tilde{x}) = 1$ and hence $\sum_{y=-\infty}^{\tilde{y}} P(y|\tilde{x}) + \sum_{y=\tilde{y}}^{\infty} P(y|\tilde{x}) = 1$ so we obtain

$$\sum_{y=-\infty}^{\tilde{y}} P(y|\tilde{x}) = \sum_{y=\tilde{y}}^{\infty} P(y|\tilde{x}) = \frac{1}{2} \Rightarrow$$
$$\tilde{y} = \text{ median of } P(y|\tilde{x}).$$

This holds true for all $x \in X$ so the Bayes estimator is

$$f^*(x) = \text{ median } P(y|x)$$

2 Question 7

2.1 Part a

We define:

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

and we wish to check for which values of $c \in \mathbb{R}$ is K_c a positive semi-definite kernel.

For $\underline{c \geqslant 0}$, let

$$\phi : \mathbb{R}^n \to \mathbb{R}^{n+1}$$

defined by

$$\phi(\boldsymbol{x}) = (\sqrt{c}, x_1, \ldots, x_n).$$

Then for any $\boldsymbol{x}, \boldsymbol{z} \in \mathbb{R}^n$,

$$K_c(\boldsymbol{x}, \boldsymbol{z}) = c + \sum_{i=1}^{n} x_i z_i =$$
$$(\sqrt{c}, x_1, \ldots, x_n)(\sqrt{c}, z_1, \ldots, z_n)^T =$$
$$< \phi(\boldsymbol{x}), \phi(\boldsymbol{z}) > .$$

In lectures we saw the theorem which states that K is positive semidefinite if and only if it can be represented as the inner product of a feature map $\phi : \mathbb{R}^n \to \mathbb{W}$, for a Hilbert space $\mathbb{W}$. Therefore, since K_c for $c \geqslant 0$, can be represented as the inner product of the feature map ϕ (as defined above) and $\mathbb{R}^{n+1}$ equipped with the dot product is a Hilbert space, then K_c is a positive semidefinite kernel for all $c \geqslant 0$.

We will now argue that K_c is <u>not</u> a positive semi-definite kernel for any $c < 0$.
If $\underline{c < 0}$, for K_c to be positive semi-definite we must have:

$$\sum_{i=1}^{m}\sum_{j=1}^{m} c_i c_j K_c(\boldsymbol{x}_i, \boldsymbol{x}_j) =$$
$$\sum_{i=1}^{m}\sum_{j=1}^{m} c_i c_j (c + \boldsymbol{x}_i \cdot \boldsymbol{x}_j) \geqslant 0$$

for all $c < 0, m \in \mathbb{N}, \boldsymbol{x}_i \in \mathbb{R}^n, c_i \in \mathbb{R}, i = 1, \ldots, m$. In other words, K_c must satisy the positive semi-definiteness property.

Since the above inequality must hold true for all such choices of m, $\boldsymbol{x}_i$ and c_i, in order to show that K_c is <u>not</u> a positive semi-definite kernel, it is sufficient to find a choice of $m, c_i, \boldsymbol{x}_i$ for which the above does not hold, for any $c < 0$.

Let $c < 0$ and let m = 2, $c_1 = c_2 = 1$, $n = 2$ and $\boldsymbol{x}_1 = (1, 0), \boldsymbol{x}_2 = (-1, 0)$. Then we have,

$$\begin{aligned}
\sum_{i=1}^{2}\sum_{j=1}^{2} c_i c_j (c + \boldsymbol{x}_i \cdot \boldsymbol{x}_j) = \\
(c + \boldsymbol{x}_1 \cdot \boldsymbol{x}_1) + 2(c + \boldsymbol{x}_1 \cdot \boldsymbol{x}_2) + (c + \boldsymbol{x}_2 \cdot \boldsymbol{x}_2) = \\
(c + 1) + 2(c - 1) + (c + 1) = \\
4c < 0
\end{aligned}$$

Hence, for any choice of $c < 0, K_c$ is not a positive semidefinite kernel.
Therefore, we conclude that K_c is a positive semi-definite kernel only for $c \geqslant 0$.

2.2 Part b

Suppose we use $K_c, c \geqslant 0$ for linear regression (least squares). As we showed in lectures (with $\lambda = 0$) we have the regression coefficients $\boldsymbol{\alpha} = K^{-1}\boldsymbol{y}$ and the regression function $f(\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i K_c(\boldsymbol{x_i}, \boldsymbol{x})$ where K is the gram matrix which in our case is equal to:

$$K_c = \begin{pmatrix} c + \boldsymbol{x}_1 \cdot \boldsymbol{x}_1 & \ldots & c + \boldsymbol{x}_1 \cdot \boldsymbol{x}_m \\ \vdots & \ddots & \vdots \\ c + \boldsymbol{x}_m \cdot \boldsymbol{x}_1 & \ldots & c + \boldsymbol{x}_m \cdot \boldsymbol{x}_m \end{pmatrix}$$

As we showed in part a, for $c \geqslant 0$ the kernel can be represented in terms of the feature map,

$$\phi : \mathbb{R}^n \to \mathbb{R}^{n+1}$$

defined by

$$\phi(\boldsymbol{x}) = (\sqrt{c}, x_1, \ldots, x_n).$$

This is essentially like adding a fixed bias term $\sqrt{c}$ to linear regression. Without c we have, $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i \cdot \boldsymbol{x}_j$ so it's equivalent to least squares without a feature map hence we want to minimise

$$\sum_i (\boldsymbol{w}^T \boldsymbol{x}_i - y)^2$$

with respect to $\boldsymbol{w} \in \mathbb{R}^n$. Working on the feature space and a adding a fixed bias term, with ϕ as defined above, i.e. $\phi(\boldsymbol{x}) = (\sqrt{c}, \boldsymbol{x})$ and $w_0 = 1$, we now want to minimise

$$\sum_i (\boldsymbol{w}^T \boldsymbol{x}_i + \sqrt{c} - y)^2$$

with respect to $\boldsymbol{w} \in \mathbb{R}^n$. By differentiating with respect to $\boldsymbol{w}$ and setting equal to 0 we see that this is achieved when:

$$\boldsymbol{w} = (X^T X)^{-1} X^T (y - \sqrt{c}).$$

We can also see this if we consider linear regression in a probabilistic context. The gram matrix is essentially the covariance function of the $y_i's$, since we believe that the y values of the training data points are generated using this covariance matrix. When $c = 0$ we have the same case as in the input space ($K_c = X^T X$) and the covariance matrix takes the form:

$$K_0 = \begin{pmatrix} \boldsymbol{x}_1 \cdot \boldsymbol{x}_1 & \ldots & \boldsymbol{x}_1 \cdot \boldsymbol{x}_m \\ \vdots & \ddots & \vdots \\ \boldsymbol{x}_m \cdot \boldsymbol{x}_1 & \ldots & \boldsymbol{x}_m \cdot \boldsymbol{x}_m \end{pmatrix}$$

We can check that all the points y are either positively or negatively correlated as correlations take the values 1 or -1, that is

$$corr(y_i, y_j) = \frac{(K_0)_{ij}}{\sqrt{(K_0)_{ii}}\sqrt{(K_0)_{jj}}} = \frac{\boldsymbol{x}_i \cdot \boldsymbol{x}_j}{\sqrt{||\boldsymbol{x}_i||^2}\sqrt{||\boldsymbol{x}_j||^2}} = sign(\boldsymbol{x}_i \cdot \boldsymbol{x}_j)(= \pm 1)$$

Now as c increases, c > 0,

$$corr(y_i, y_j) = \frac{c + \boldsymbol{x}_i \cdot \boldsymbol{x}_j}{\sqrt{c + ||\boldsymbol{x}_i||^2}\sqrt{c + ||\boldsymbol{x}_j||^2}} \to 1 \text{ as } c \to \infty.$$

In other words, as c increases, all $y_i's$ become positively correlated with correlation equal to 1, so we

believe that for a new point $\boldsymbol{x}$, we would predict $y = f(\boldsymbol{x})$ positively correlated with all y's in our data set. This is essentially the same as enforcing a very large bias term to our linear regression.

3 Question 8

We have the data set $(\boldsymbol{x}_1, y_1), \ldots (\boldsymbol{x}_m, y_m) \in \mathbb{R}^n \times \{-1, 1\}$ and want to perform linear regression to train a classifier on this data set. Hence, we first need an estimate $f(\boldsymbol{x}_i) \in \mathbb{R}$, such that $sign(f(\boldsymbol{x}_i)) = sign(y_i)$, for each $\boldsymbol{x}_i$ on the data set. Given these estimates we can then perform linear regression (least squares), hence wishing to minimise:

$$\epsilon(\boldsymbol{w}) = \sum_{i=1}^{m} (f(\boldsymbol{x}_i) - \boldsymbol{w}^T \boldsymbol{x}_i)^2$$

By differentiating with respect to $\boldsymbol{w}$ and setting equal to zero we obtain:

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{w}} = \sum_{i=1}^{m} -2\boldsymbol{x}_i^T (f(\boldsymbol{x}_i) - \boldsymbol{w}^T \boldsymbol{x}_i) = \\ -2X^T(\boldsymbol{f}(\boldsymbol{x}) - X\boldsymbol{w}) = 0 \\ \Rightarrow \boldsymbol{w} = (X^T X)^{-1} X^T \boldsymbol{f}(\boldsymbol{x}) \end{aligned}$$

where $\boldsymbol{f}(\boldsymbol{x})$ is the column vector $(f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_m))^T$.

Considering this in the feature space, and using the representer theorem, as showed in lectures, we have

$$\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i \boldsymbol{\phi}(\boldsymbol{x}_i) \Rightarrow f(\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i K_\beta(\boldsymbol{x}, \boldsymbol{x}_i)$$

where $\alpha = K_\beta^{-1} \boldsymbol{f}(\boldsymbol{x})$ and $(K_\beta)_{i,j} = K_\beta(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

So for a new data point $\boldsymbol{t}$ we classify $\boldsymbol{t}$ as $sign(f(\boldsymbol{t}))$.

Now let

$$\begin{aligned} k = \underset{i=1,\ldots,m}{\arg\min} ||\boldsymbol{t} - \boldsymbol{x}_i||, \text{ i.e.} \\ ||\boldsymbol{t} - \boldsymbol{x}_k|| < ||\boldsymbol{t} - \boldsymbol{x}_i|| \forall i \in 1, \ldots, m. \end{aligned} \tag{1}$$

or in other words, let $\boldsymbol{x}_k$ be the nearest neighbour to $\boldsymbol{t}$.

Expanding $sign(f(\boldsymbol{t}))$ we have:

$$sign(f(\boldsymbol{t})) = sign(\sum_{i=1}^{m} \alpha_i K_\beta(\boldsymbol{t}, \boldsymbol{x}_i)) =$$
$$sign(\sum_{i=1}^{m} \alpha_i e^{-\beta||\boldsymbol{t}-\boldsymbol{x}_i||^2}) =$$
$$sign(\frac{\sum_{i=1}^{m} \alpha_i e^{-\beta||\boldsymbol{t}-\boldsymbol{x}_i||^2}}{e^{-\beta||\boldsymbol{t}-\boldsymbol{x}_k||^2}})$$

since dividing by a positive number doesn't influence the sign.

Then

$$sign(\frac{\sum_{i=1}^{m} \alpha_i e^{-\beta||\boldsymbol{t}-\boldsymbol{x}_i||^2}}{e^{-\beta||\boldsymbol{t}-\boldsymbol{x}_k||^2}}) =$$

$$sign(\alpha_k + \sum_{i \in \{1,\dots,m\} \setminus \{k\}} \alpha_i e^{-\beta(||\boldsymbol{t}-\boldsymbol{x}_i||^2 - ||\boldsymbol{t}-\boldsymbol{x}_k||^2)}$$

From (1), we have that $||\boldsymbol{t}-\boldsymbol{x}_i|| - ||\boldsymbol{t}-\boldsymbol{x}_k|| < 0 \ \forall i \in 1, \dots, m$, so taking the limit $\beta \to \infty$ we get $e^{-\beta(||\boldsymbol{t}-\boldsymbol{x}_i||^2 - ||\boldsymbol{t}-\boldsymbol{x}_k||^2)} \to 0$ and hence

$$\alpha_k + \sum_{i \in \{1,\dots,m\} \setminus \{k\}} \alpha_i e^{-\beta(||\boldsymbol{t}-\boldsymbol{x}_i||^2 - ||\boldsymbol{t}-\boldsymbol{x}_k||^2)} \to \alpha_k \text{ as } \beta \to \infty.$$

Hence,

$$sign(f(\boldsymbol{t})) \to sign(\alpha_k)$$

But as $\beta \to \infty$, the gram-matrix K tends to the identity matrix since:

$$K_\beta = \begin{pmatrix} e^{-\beta||\boldsymbol{x}_1-\boldsymbol{x}_1||^2} & e^{-\beta||\boldsymbol{x}_1-\boldsymbol{x}_2||^2} & \dots & e^{-\beta||\boldsymbol{x}_1-\boldsymbol{x}_m||^2} \\ \vdots & \ddots & \vdots & \\ e^{-\beta||\boldsymbol{x}_m-\boldsymbol{x}_1||^2} & e^{-\beta||\boldsymbol{x}_m-\boldsymbol{x}_2||^2} & \dots & e^{-\beta||\boldsymbol{x}_m-\boldsymbol{x}_m||^2} \end{pmatrix} =$$

$$\begin{pmatrix} 1 & e^{-\beta||\boldsymbol{x}_1-\boldsymbol{x}_2||^2} & \dots & e^{-\beta||\boldsymbol{x}_1-\boldsymbol{x}_m||^2} \\ \vdots & \ddots & \vdots & \\ e^{-\beta||\boldsymbol{x}_m-\boldsymbol{x}_1||^2} & e^{-\beta||\boldsymbol{x}_m-\boldsymbol{x}_2||^2} & \dots & 1 \end{pmatrix} \rightarrow$$

$$\begin{pmatrix} 1 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \\ 0 & \dots & 0 & 1 \end{pmatrix}$$

$$= I_{m\times m}$$

Therefore,

$$\boldsymbol{\alpha} = K_\beta^{-1}\boldsymbol{f}(\boldsymbol{t}) \rightarrow I_{m\times m}^{-1}\boldsymbol{f}(\boldsymbol{t}) = \boldsymbol{f}(\boldsymbol{t})$$

as $\beta \rightarrow \infty$.

Hence,

$$sign(f(\boldsymbol{t})) \rightarrow sign(\alpha_k) \rightarrow sign(f(\boldsymbol{x}_k)) = sign(y_k)$$

as $\beta \rightarrow \infty$.

We proved that as $\beta \rightarrow \infty$ the classifier tends to the desirable classification. Therefore, we need to choose β large enough such as the required terms are approximately equal to 0. Let d_{min} be the minimum distance between any two points of the dataset, i.e.

$$d_{min} = \min_{\boldsymbol{x}_i,\boldsymbol{x}_j}(||\boldsymbol{x}_i - \boldsymbol{x}_j||),$$

let $\tilde{d}^2_{min}$ be defined as

$$\tilde{d}^2_{min} = \min_{\boldsymbol{x}_i}(||\boldsymbol{t} - \boldsymbol{x}_i||^2 - ||\boldsymbol{t} - \boldsymbol{x}_k||^2)$$

and let $d^2 = \min\{d^2_{min}, \tilde{d}^2_{min}\}$.

Then if we take β large enough such that

$$e^{-\beta d^2} \cong 0$$

then

$$e^{-\beta(||\boldsymbol{x}_i - \boldsymbol{x}_j||^2)} \leqslant e^{-\beta d^2} \cong 0$$

for all $i = 1, \ldots, m$ and hence $K \cong I$ and

$$e^{-\beta(||\boldsymbol{t} - \boldsymbol{x}_i||^2 - ||\boldsymbol{t} - \boldsymbol{x}_k||^2)} \leqslant e^{-\beta \tilde{d}^2} \cong 0$$

for all $i = 1, \ldots, m$ and hence $sign(f(\boldsymbol{t})) \cong sign(y_k)$, as we shown above, and we obtain the same classification as the 1- nearest neighbour, that is for every new data point $\boldsymbol{t}$, class$(\boldsymbol{t}) = sign(y_k)$ where $\boldsymbol{x}_k$ is the nearest neighbour of $\boldsymbol{t}$.

4 Question 9

For this question, the idea of the algorithm was inspired by `https://people.sc.fsu.edu/~jburkardt/classes/imps_2017/11_28/2690705.pdf` and `https://www.xarg.org/2018/07/lightsout-solution-using-linear-algebra/`.

We represent the board as an $n \times n$ matrix with entries equal to 0 and 1, where a 0 entry denotes a hole with a hidden mole and a 1 entry denotes a hole with a visible mole. Let I be the matrix representing the initial configuration. Then everytime we wack a mole we change that entry and its adjacent ones, that is we change it to 0 if it was 1 and to 1 if it was 0. This is equivalent to adding 1 $\mod 2$ to these entries, since $1 + 1 = 2 \equiv 0(\mod 2)$ and $0 + 1 = 1(\mod 2)$. So if for example we have a 4×4 board, with initial configuration I, and we hit the hole corresponding to entry $I_{2,3}$ with the mallet, it's like we are adding ($\mod 2$) the matrix

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

to the matrix I. Going back to the general $n \times n$ case, call P_{ij} every such matrix, corresponding to the effect of hitting hole ij.

The order in which we are hitting the holes doesn't matter since addition is commutative. Also, hitting a hole twice results to no effect as the actions result to adding $1 + 1 = 0 \mod 2$ to that hole and its adjacent ones. Hence, it suffices to find out which holes we should hit once and which ones

not at all. As explained above, this is equivalent to the problem of finding out which of the matrices P_{ij}, for all possible i,j pairs, we should add to matrix I in order to get the zero $n \times n$ matrix. So suppose we introduce the coefficients $c_{ij} \in \{0, 1\}$, each corresponding to P_{ij}. Then it suffices to find out which ones should be equal to 0 and which ones should be equal to 1 such that:

$$\begin{aligned} I + \sum_{i,j} c_{ij} P_{ij} &= 0_{n\times n} (\mod 2) \Rightarrow \\ I &= -\sum_{i,j} c_{ij} P_{ij} (\mod 2) \Rightarrow \\ I &= \sum_{i,j} c_{ij} P_{ij} (\mod 2) \end{aligned} \tag{2}$$

since $-1 \equiv 1 (\mod 2)$.

We can think of this expression in a vectorised form, where $\boldsymbol{I'}$ is an $n^2 \times 1$ vector with entries equal to the matrix entries of I, P is a $n^2 \times n^2$ matrix equal to

$$\begin{pmatrix} (P_{11})_{1,1} & \dots & (P_{nn})_{1,1} \\ (P_{11})_{1,2} & \dots & (P_{nn})_{1,2} \\ \vdots & \ddots & \vdots \\ (P_{11})_{n-1,n} & \dots & (P_{nn})_{n-1,n} \\ (P_{11})_{n,n} & \dots & (P_{nn})_{n,n} \end{pmatrix}$$

and $\boldsymbol{c}$ is a $n^2 \times 1$ vector with entries equal to c_{ij}, i.e. $\boldsymbol{c} = (c_{11}, \dots, c_{1n}, c_{21}, \dots, c_{2n}, \dots, c_{n1}, \dots, c_{nn})$. Then $\boldsymbol{I'} = P\boldsymbol{c}$ is a set of n^2 equations with n^2 unknowns, where the unknowns are the $c_{ij}, i, j \in \{1, \dots, n\}$ and hence can be solved using matrix inversion of an $n^2 \times n^2$ matrix or by Gaussian elimination. All such methods are polynomial in n. However, not always such a solution exists as this depends on whether the matrix P is invertible, so the algorithm will return a solution if such one exists.

To conclude, the algorithm takes as input the initial configuration I and gives as output a sequence $s = ((ij))$ that solves the whack-a-mole game. To complete this task, the algorithm solves the system of equations (2) and returns s where $(ij) \in s \iff c_{ij} = 1$.