Supervised Learning CW1

Jinge Wu, Shiqi Su

November 16, 2020

Part A

Linear Regression

Basic idea of regression is that we want to find a function $g(\mathbf{x})$ through a set of samples $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_l, y_l)\}$. Thus, we look for a function g such that $y_i \simeq g(\mathbf{x}_i)$ for all $i = 1, 2, ..., l$.

The points $\mathbf{x}_i$ along with their given label y_i make up the training set to estimate the regression function g. And we use the term test set to denote the set of test points $\mathbf{x}_{test,i}$ on which we afterwards make predictions.

In linear regression, the functional relation we are looking for is of the form:

$$y_i \simeq g(\mathbf{x}_i) = {\mathbf{x}_i}^T \mathbf{w} \tag{1}$$

where $\mathbf{w}$ is called the weight vector.

In Least Squares Regression (LSR), in order to try to achieve that $g(\mathbf{x}_i) = {\mathbf{x}_i}^T \mathbf{w}$, we look for the weight vector that minimizes the mean of the squared errors on all training samples:

$$\mathbf{w}^* = \arg\max_w \frac{1}{l} \sum_{i=1}^{l} ({\mathbf{x}_i}^T \mathbf{w} - y_i)^2 \tag{2}$$

Using the notation X to be the $l \times k$ matrix

$$X = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,k} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ x_{l,1} & x_{m,2} & \cdots & x_{l,k} \end{pmatrix} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_k) \tag{3}$$

and the optimal solution of weight vector $\mathbf{w}^*$ is

$$\mathbf{w}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \tag{4}$$

for non-singular $\mathbf{X}'\mathbf{X}$. In linear regression with basis functions we fit the data sequence with a linear combination of basis Φ, which is a $l \times k$ matrix, where

$$\Phi := \begin{pmatrix} \phi(\mathbf{x}_1) \\ \vdots \\ \phi(\mathbf{x}_l) \end{pmatrix} = \begin{pmatrix} \phi_1(\mathbf{x}_1) & \cdots & \phi_k(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ \phi_1(\mathbf{x}_l) & \cdots & \phi_k(\mathbf{x}_l) \end{pmatrix} \tag{5}$$

Linear regression on the transformed dataset thus finds a k-dimensional vector $\mathbf{w} = (w_1, ..., w_k)$ such that

$$(\Phi\mathbf{w} - \mathbf{y})^T (\Phi\mathbf{w} - \mathbf{y}) = \sum_{t=1}^{l} (y_t - \sum_{i=1}^{k} w_i \phi_i(x_t))^2 \tag{6}$$

is minimized. A common basis used is the polynomial basis $\{\phi_1(x) = 1, \phi_2(x) = x, \phi_3(x) = x^2, \phi_4(x) = x^3, .., \phi_k(x) = x^{k-1}\}$ of dimension k (order $k-1$).

Question1

This question applied the polynomial basis $\{\phi_1(x) = 1, \phi_2(x) = x, \phi_3(x) = x^2, \phi_4(x) = x^3, .., \phi_k(x) = x^{k-1}\}$ of dimension k (order $k-1$) for linear regression with a data set of $\{(1, 3), (2, 2), (3, 0), (4, 5)\}$.

(a) For each of the polynomial bases of dimension $k = 1, 2, 3, 4$ fit the data set. Below are the figures plotted by Matlab.

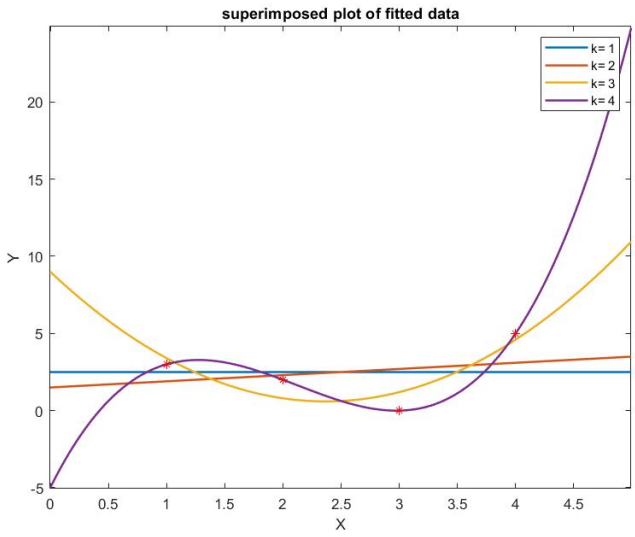

Figure 1: 4 fitted curves

From Figure 1, we can observe the fitted curve and data points. It is clear that, $k = 4$ give the best fit. Almost all the data points are on the fitted curve.

(b) Give the equations corresponding to the curves fitted for $k = 1, 2, 3, 4$. The regression equations for $k = 1, 2, 3, 4$ can be expressed as:

- $k = 1, y_i \simeq g(\mathbf{x}_i) = 2.5$
- $k = 2, y_i \simeq g(\mathbf{x}_i) = 1.5 + 0.4\mathbf{x}_i$
- $k = 3, y_i \simeq g(\mathbf{x}_i) = 9 - 7.1\mathbf{x}_i + 1.5{\mathbf{x}_i}^2$
- $k = 4, y_i \simeq g(\mathbf{x}_i) = -5 + 15.1667\mathbf{x}_i - 8.5{\mathbf{x}_i}^2 + 1.3333{\mathbf{x}_i}^3$

(c) For each fitted curve $k = 1, 2, 3, 4$, give the mean square error. We can use mean squared error (MSE) to measure the error

$$MSE = \frac{1}{l}\sum_{i=1}^{l}(y_i - \mathbf{w} \cdot \mathbf{x}_i)^2 \tag{7}$$

The MSE calculated from Matlab is

- $k = 1$, MSE = 3.25
- $k = 2$, MSE = 3.05
- $k = 3$, MSE = 0.80
- $k = 4$, MSE = 7.896e-31

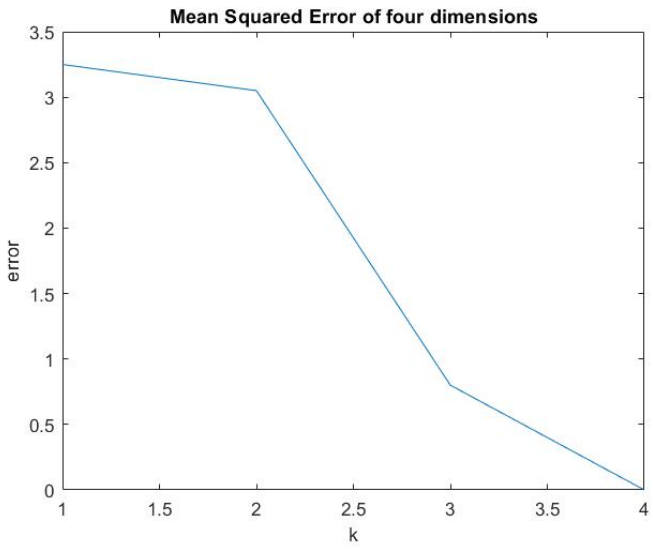

Figure 2: MSE plot for each fitted curve

Also it is observed that the error is decreasing when the dimension increases. When $k = 4$, MSE is almost zero, which indicates the best fit in the chosen polynomial basis.

Question2

(a) Define

$$g_\sigma(x) = sin^2(2\pi x) + \epsilon \tag{8}$$

where ϵ is a random variable distributed normally with mean 0 and variance σ^2. Thus $g_\sigma(x)$ is a random function such that $sin^2(2\pi x)$ is computed and then the normal noise is added. We then sample x_i uniformly at random from the interval [0, 1] 30 times creating $(x_1, ..., x_{30})$ and apply $g_{0.07}$ to each x creating the data set

$$S_{0.07,30} = \{(x_1, g_{0.07}(x_1)), ..., (x_{30}, g_{0.07}(x_{30})\} \tag{9}$$

as the training data. Below is the plot of simulated data.

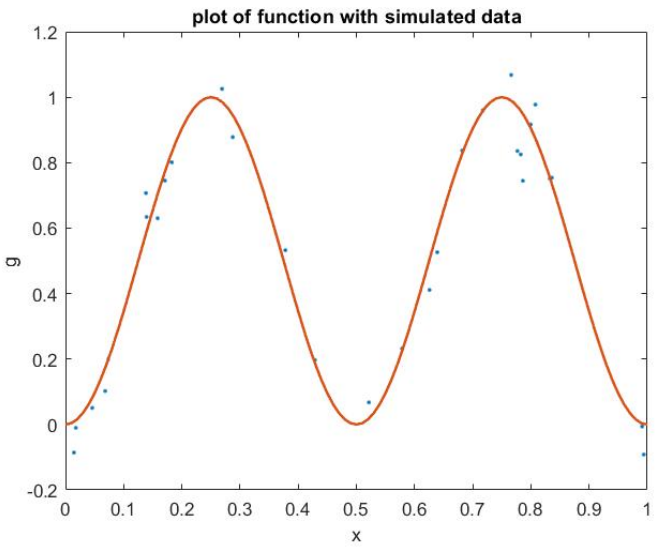

Figure 3: Simulated data with the function $sin^2(2\pi x)$

Then we fit the data set with a polynomial basis of dimension $k = 2, 5, 10, 14, 18$. Here are the plots of these 5 curves superimposed over a plot of data points.

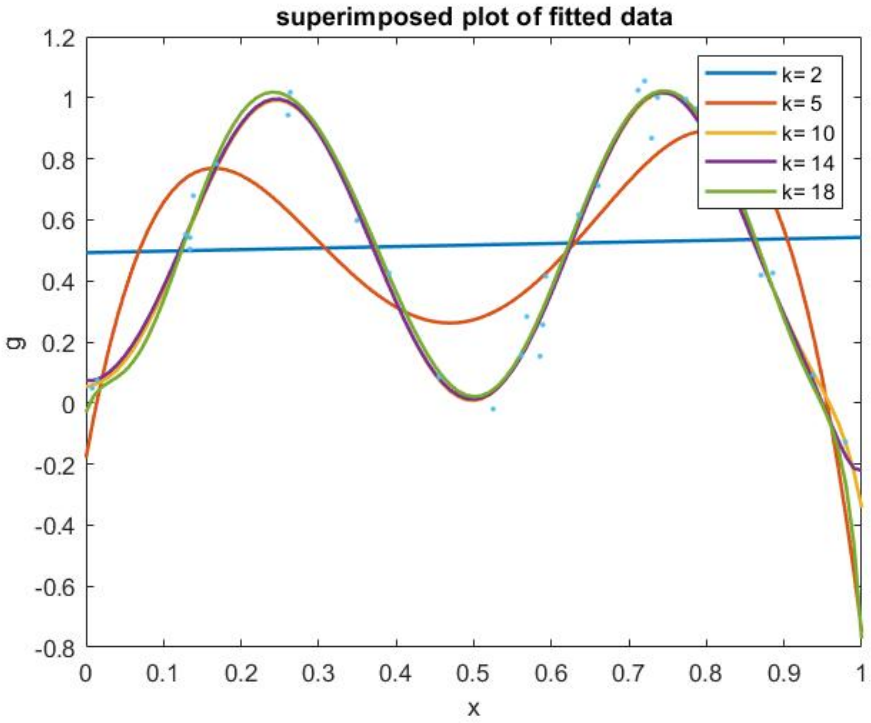

Figure 4: Data set fitted with dimension $k = 18$

We find that the fitted curves become similar to g_σ when $k > 5$. Moreover, the fitted curve becomes closer to the data points as k increases.

(b) Calculate the training error $tse_k(S)$, which is the MSE of the fitting of the data set S with polyno-

mial basis of dimension k. And plot the natural log of the training error versus the polynomial basis of dimension $k = 1, ..., 18$.

The logarithm of MSEs are plotted below:

Figure 5: natural log(ln) of the training error versus the polynomial dimension $k = 1, 2, ..., 18$

From Figure 5, it is obvious that this is a decreasing function and has the minimum value when $k = 18$.

(c) Generate a test set T of a thousand points,

$$T_{0.07,1000} = \{(x_1, g_{0.07}(x_1)), ..., (x_{1000}, g_{0.07}(x_{1000})\} \tag{10}$$

Calculate the test error $tse_k(S, T)$, which is the MSE of the test set T on the polynomial of dimension k fitted from training set S.

After generating the test data, the test error can be plotted using Matlab, and here is the figure:

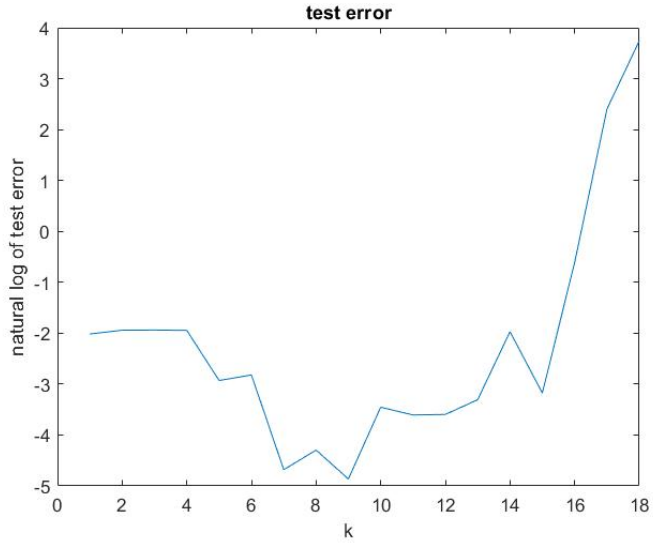

Figure 6: natural log(ln) of the test error versus the polynomial dimension $k = 1, 2, ..., 18$

In Figure 6, unlike the training error this is not a decreasing function. This is the phenomena of over-fitting. Although the training error decreases with growing k the test error eventually increases since rather than fitting the function, we begin to fit to the noise.

(d) Repeat the previous part but instead of plotting the results of a single "run", plot the average results of a 100 runs.

Figure 7 and 8 show the results.

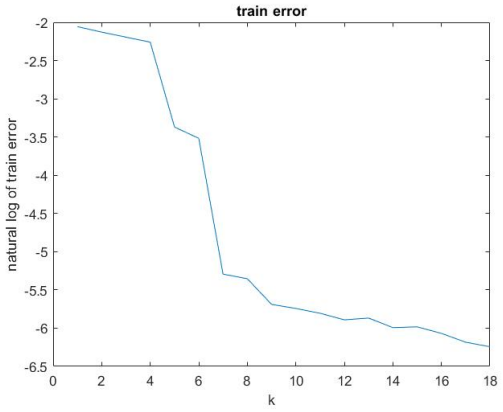

Figure 7: natural log(ln) of the average training error in 100 runs

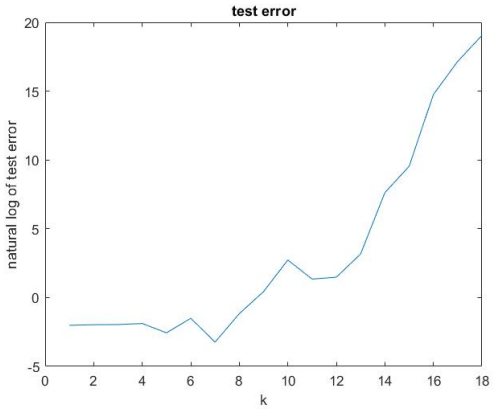

Figure 8: natural log(ln) of the average test error in 100 runs

Comparing to the previous process, the results of running 100 times stay the same as before, where it still indicates the phenomena of over-fitting. Besides, the graphs are more smoother than a single run.

Question3

Now use basis (for $k = 1, ..., 18$)

$$sin(1\pi x), sin(2\pi x), sin(3\pi x), ..., sin(k\pi x)$$

And repeat the experiments in Question 2 with the above basis.

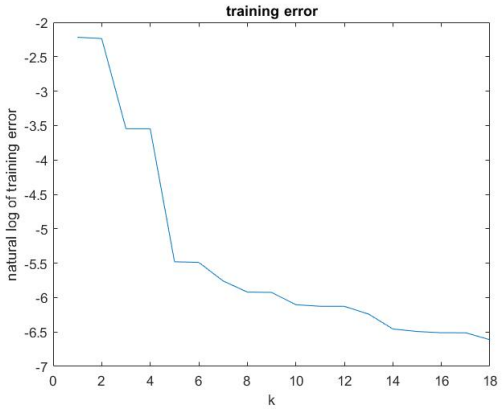

Figure 9: natural log(ln) of the average training error in one run

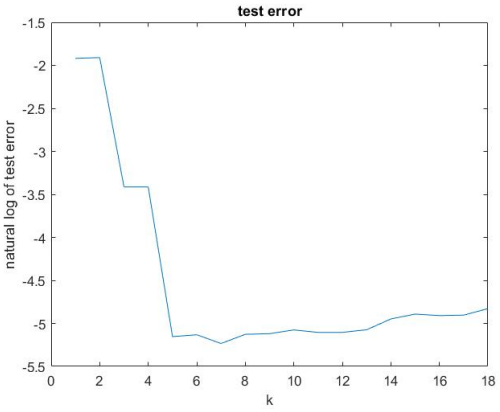

Figure 10: natural log(ln) of the test error in one run

After switching the basis function, the results remains the same as Question 2, which show the phenomena of over-fitting. In Figure 10, it can be seen that the test error is slightly larger after $k = 5$.

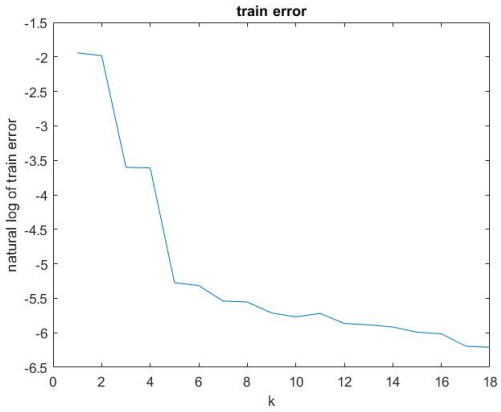

Figure 11: natural log(ln) of the average training error in 100 runs

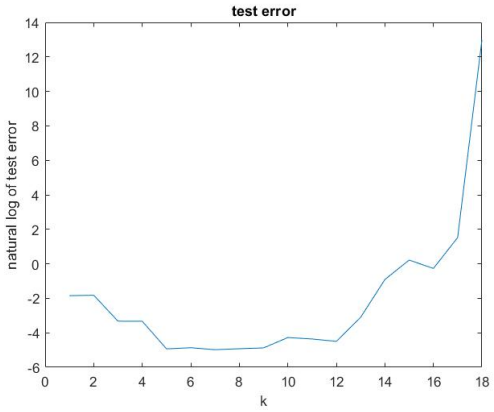

Figure 12: natural log(ln) of the average test error in 100 runs

After running 100 times, we can observe that the test error gets dramatically large as k increases. One possible reason is that, the difference between the number of training data and the number of test data get greater after running 100 times.

Question4

Boston housing is a classic data set where we will use 12 attributes to predict the 13th. There are 506 entries which we will split into train and test. The training set is 2/3 of the data set, and the test set is 1/3 of the data set. In the following average the results over 20 runs (each run based on a different (2/3,1/3) random split).

(a)Predicting with the mean y-value.

This part performs a naive Regression. We create a vector of ones that is the same length as the training set using the function *ones*. Do the same for the test set. By using these vectors we will be fitting the data with a constant function. By performing linear regression on the training set, the fitted curve is plotted below:

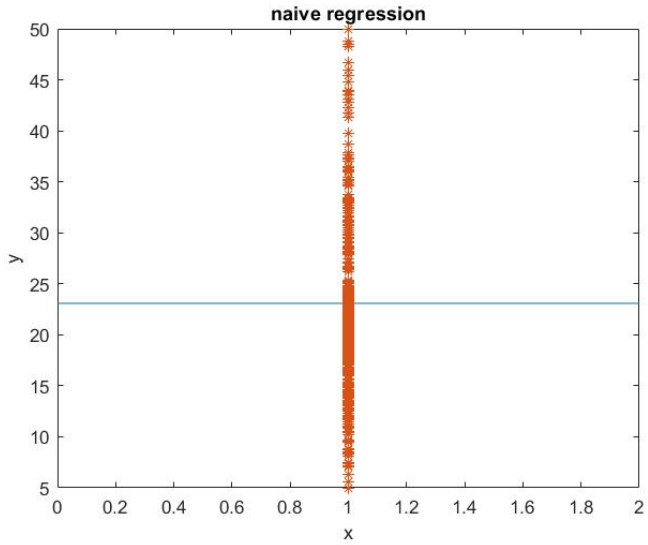

Figure 13: Naive Regression using function ones, the fitted curve $y = 22.2386$.

In this case, the fitted curve is a constant function $y = 22.2386$. The MSE of naive regression is that $MSE_{train} = 84.3907 \pm 5.4691$, $MSE_{test} = 84.0561 \pm 4.9798$. The interpretation of the constant function is that those attributes do not contribute to the housing price prediction (it means that y is not related with x any more).

(b) Predicting with a single attribute.

In this case, we regression the model with one attribute at one time. For each of the twelve attributes, perform a linear regression using only the single attribute but incorporating a bias term so that the inputs are augmented with an additional 1 entry, $(\mathbf{x}_i, 1)$, so that we learn a weight vecto r$\mathbf{w} \in \Re^2$.

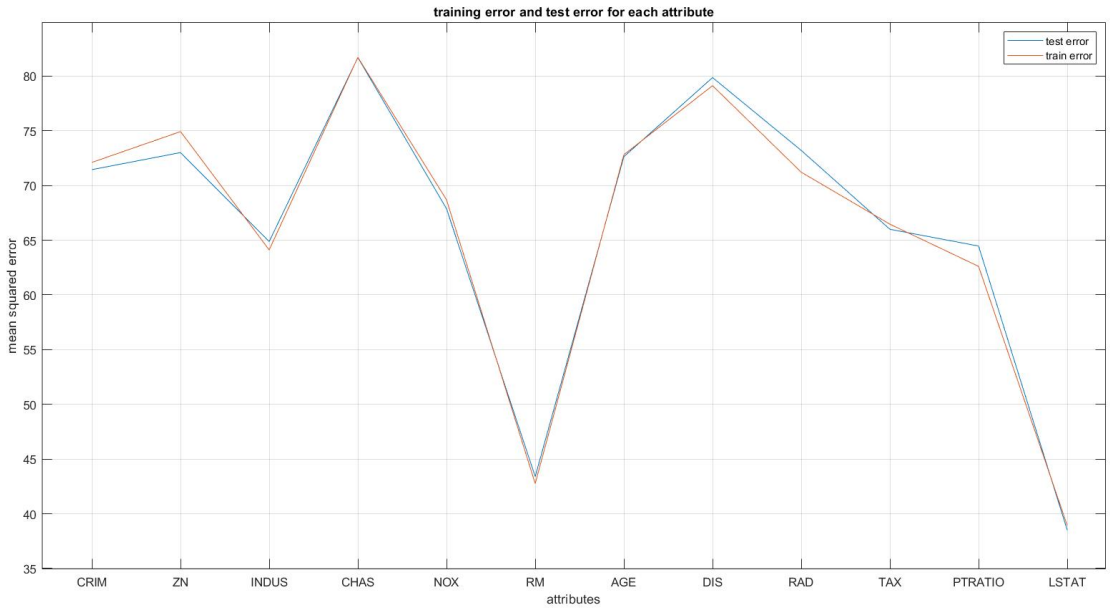

Figure 14: Linear regression with single attributes.

In Figure 14, it can be seen that different attribute has different training errors and test errors, where the attribute 12 ("LSTAT") obtains the least error and the attribute 4 ("CHAS") achieves the greatest error. More detailed MSE can be find in Table 1.

(c) Predicting with all the attributes.

Now we would like to perform linear regression using all of the data attributes at once. Perform linear regression on the training set using this regressor, and incorporate a bias term as above. From the results from Matlab, we find that this method outperforms any of the individual regressors with $MSE_{train} = 21.6283 \pm 1.908$ and $MSE_{test} = 23.2726 \pm 1.7815$. The MSE of linear regression using all attributes achieves the least error.

Question5

This question considers Kernelised ridge regression. A commonly used kernel function for which this is the case is the Gaussian kernel, which is defined as:

$$K(\mathbf{x}_i, \mathbf{x}_j) = exp(-\frac{\| \mathbf{x}_i - \mathbf{x}_j \|^2}{2\sigma^2}) \tag{11}$$

The dual weight vector $\alpha^* \in \Re^n$

$$\alpha^* = (\mathbf{K} + \gamma l \mathbf{I}_l)^{-1}\mathbf{y} \tag{12}$$

The evaluation of the regression function on a test point can be reformulated as:

$$y_{test} = \sum_{i=1}^{l} {\alpha_i}^* K(\mathbf{x}_i, \mathbf{x}_{test}) \tag{13}$$

where the $\mathbf{K}$ is the kernel function $\mathbf{K}_{test} = K(\mathbf{x}_i, \mathbf{x}_{test})$. The cost function (MSE) can then be calculated as follows:

$$\frac{1}{l}(\mathbf{K}_{test}\alpha - \mathbf{y})^T(\mathbf{K}_{test}\alpha - \mathbf{y}) \tag{14}$$

(a)Create a vector of γ values $[2^{-40}, 2^{-39}, ..., 2^{-26}]$and a vector of σ values $[2^7, 2^{7.5}, ..., 2^{12.5}, 2^{13}]$. Perform kernel ridge regression on the training set using five-fold cross-validation to choose among all pairing of the values of γ and σ. Choose the γ and σ values that perform the best to compute the predictor (by then retraining with those parameters on the training set) that you will use to report the test and training error.

By running codes in Matlab, the best choice is $\gamma = 1.819e-12$, $\sigma = 2048$, with train error =7.7467, test error = 11.0395.

(b) Plot the "cross-validation error" (mean over folds of validation error) as a function of γ and σ. The cross-validation error is defined as the cost function F, which is the average of each training error.

$$F = \frac{1}{m}(\mathbf{K}_{train}\alpha - \mathbf{y}_{train})^T(\mathbf{K}_{train}\alpha - \mathbf{y}_{train}) \tag{15}$$

where α can be calculated using Equation 12 and $m = 5$ in this part.

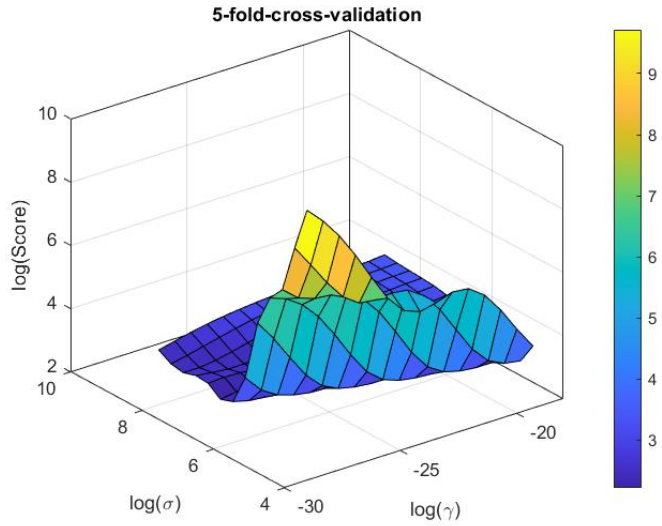

Figure 15: Linear regression with single attributes.

Figure 15 show the error with different choice of γ and σ, where the log(score) is the logarithm of cost function F. It can be seen that generally speaking, choice of smaller γ and larger σ gives less score, which means less validation error.

(c) Calculate the MSE on the training and test sets for the best γ and σ. Using the best choice in (a), we get MSE of training and test sets:$MSE_{train} = 8.3811$, $MSE_{test} = 9.1236$.

(d)Repeat the experiment over 20 random (2/3, 1/3) splits of data. Record the train/test error and the standard deviation of the train/test errors and summarise these results in the following type of table:

Method	MSE train	MSE test
Naive Regression	84.3907 ±5.4691	84.0561± 4.9798
Linear Regression (attribute 1)	72.1110 ±5.8049	71.4531 ± 3.8864
Linear Regression (attribute 2)	74.9309 ±4.2997	73.0132 ±5.3281
Linear Regression (attribute 3)	64.1157 ±3.9371	64.8816 ±6.0147
Linear Regression (attribute 4)	81.7211 ±5.0331	81.7009 ± 4.5021
Linear Regression (attribute 5)	68.6950 ±4.4203	67.8790 ± 5.4073
Linear Regression (attribute 6)	42.7640 ±4.8534	43.3996 ± 4.0421
Linear Regression (attribute 7)	72.8217 ±5.1371	72.6358 ± 4.9910
Linear Regression (attribute 8)	79.1275 ±4.5948	79.8677 ± 6.6896
Linear Regression (attribute 9)	71.2181 ±5.1004	73.2099 ± 6.2625
Linear Regression (attribute 10)	66.4683 ±5.0773	66.0030 ± 5.0438
Linear Regression (attribute 11)	62.6153 ±4.0605	64.4737 ±4.0703
Linear Regression (attribute 12)	38.9380 ±2.1535	38.5125 ± 2.3396
Linear Regression (all attributes)	21.6283 ±1.908	23.2726±1.7815
Kernel Ridge Regression	8.0513 ±2.2721	11.0202 ±1.9747

Table 1: Summaries for the Kernel Ridge Regression and Linear Regression of Boston Housing

This table illustrates the training error and test error of the linear regressions by 20 runs. From the table we can find that KRR performs best with the least error and the naive regression performs worst with the greatest error.

Part B

Question 6

(a) Bayes estimator.

Define the loss function $L(y, \hat{y})$ as cost of making an estimate $\hat{y}$ if the true value is y. Then the given loss function can be regarded as an indicator, noted as $\mathbb{I}_s$ on measurable set S given measure P. Explicitly, function is written as,

$$\mathbb{I}_s = \begin{cases} 0 & \text{if } \hat{y} = y \\ c_y & \text{if } \hat{y} \neq y \end{cases}$$

From the definition of Bayes estimator $E[L(y, \hat{y})] = \underset{\hat{y}}{argmin} \int_S L(y, \hat{y}) P(y|x)\, dy$, where L_F is the loss function and $P(y|x)$ is the posterior probability.

Then w.r.t given loss function, the Bayes estimator can be

$$\begin{aligned} \min_{\hat{y}} \mathbb{E}[L(y, \hat{y})|x] &= \int_S L(y, \hat{y}) P(y|x)\, dy \\ &= \int_S \mathbb{I}_s \times C_y dP \text{ (by integration on indicator function)} \\ &= C_y P(S) \\ &= C_y \times [1 - P(y = \hat{y}|x)] \end{aligned}$$

Minimizing the $\mathbb{E}[L(y,\hat{y})|x]$ is equivalent to maximizing the correct classification. i.e Bayes estimator $f^*(x) = \max_y c_y P(y=\hat{y}|x)$

(b) Given a strictly convex differentiable function F:$\Re \to \Re$ and loss function $L_F(y,\hat{y}) := F(\hat{y}) - F(y) + (y-\hat{y})F'(y)$.

(i) If $F(x) = x^2$ then $F^{'}(x) = 2x$

$$\begin{aligned} L(y,\hat{y}) &= \hat{y}^2 - y^2 + (\hat{y}-y)\times 2y \\ &= (\hat{y}-y)^2 \end{aligned}$$

Seeing this form, the F-loss is squared loss.

(ii) In oder to prove $y = \hat{y} \leftrightarrow L(y,\hat{y}) = 0$, we have to prove necessary and efficiency separately.

- $y = \hat{y} \Rightarrow L(y,\hat{y}) = 0$ is easy.$y = \hat{y} \Rightarrow L(y,\hat{y}) = 0$ is easy. If $y = \hat{y}$ then $L_F(y,\hat{y}) = F(\hat{y}) - F(y) + 0 = 0$
- $L(y,\hat{y}) = 0 \Rightarrow y = \hat{y}$. i.e,

$$F(\hat{y}) - F(y) + (\hat{y}-y)F'(y) = 0 \tag{16}$$

By Taylor expansion and Mean Value Theorem at true point $\hat{y}$,the formula tells us,

$$F(\hat{y}) = F(y) + F'(y)(\hat{y}-y) + \frac{1}{2}F''(z)(\hat{y}-y), for z \in (y,\hat{y}) \tag{17}$$

Comparing Equation (16) and (17) the only difference is $\frac{1}{2}F''(z)(\hat{y}-y)$, which has to be zero. Because F is strictly differentiable convex, by property of convexity $F''(x) > 0$. Then the only option is $\hat{y} - y = 0 \Rightarrow \hat{y} = y$

(iii) Prove $L_F(y,\hat{y}) \geq 0$.

Since F is strictly convex, then by definition:

$$\begin{aligned} \lambda F(\hat{y}) + (1-\lambda)F(y) &\geq F(\lambda\hat{y} + (1-\lambda)y) \\ F(\hat{y}) - F(y) &\geq \frac{F(y+\lambda(\hat{y}-y))}{\lambda} \end{aligned}$$

Take limits of $\lambda \to 0$, then right hand side

$$\lim_{\lambda\to 0} \frac{F(y+\lambda(\hat{y}-y))}{\lambda} \overset{L'Hôpital'srule}{=} \lim_{\lambda\to 0} \frac{F'(y+\lambda(\hat{y}-y))\times(\hat{y}-y)}{1} = \lim_{\lambda\to 0} F'(y)(\hat{y}-y)$$

Hence equation can be $F(\hat{y}) - F(y) \geq F'(y)(\hat{y}-y)$, i.e $F(\hat{y}) - F(y) + (y-\hat{y})F'(y) \geq 0$

(iv) Derive Bayes estimator for F-loss.

From definition of Bayes estimator, we have to minimize the expected loss of this loss function.

$$f^* = \underset{\hat{y}}{argmin}\, E[L(y,f(x)] = \int_\chi \int_Y L_F(y,f(x)) dP(y|x)\, dx$$

set x=x' to be a specific number, so $dP(x')$ is a specific number. Let $f(x') = z$.

$$\epsilon(z) = \int_y L_F(y,z) dP(y|x')\, dx' = \int_y \{F(z) - F(y) + (y-z)F'(y)\} dP(y|x')\, dx'$$

Take derivative with respect to z and set it to zero,

$$\begin{aligned} \frac{\partial \epsilon(z)}{\partial x_1} &= \int_Y \{F'(z) - F'(y)\} dP(y|x')\, dx' = 0 \\ &\Longrightarrow \int_Y \{F'(z) - F'(y)\} dP(y|x') = 0 \\ \int_Y F'(z) dP(y|x') &= \int_Y F'(y) dP(y|x') \end{aligned}$$

Because $\int_Y dP(y|x') = 1$,then $F'(z) = \int_Y F'(y) dP(y|x') = E[F'(y)|x']$. Hence we can find

$$F'(f^*(x')) = \int_Y F'(y) dP(y|x) = E[F'(y)|x']$$

Question 7

(a) Kernel modification.

$K_c(\mathbf{x}, \mathbf{z}) = c + \sum_{i=1}^{n} x_i z_i = c + \mathbf{x}^T \mathbf{z}$. In order to let function K_c to be a positive semidefinite kernel, then we need to choose value c which makes the corresponding matrix $\mathbf{K}$ to be positive semidefinite.

Firstly, write down the matrix $\mathbf{K}$ with entries $K(x_i, x_j) = c + \mathbf{x_i}^T \mathbf{x_j}$,

$$K_{m,n} = \begin{pmatrix} c + \mathbf{x_1 x_1} & c + \mathbf{x_1 x_2} & \cdots & c + \mathbf{x_1 x_n} \\ c + \mathbf{x_2 x_1} & c + \mathbf{x_2 x_2} & \cdots & c + \mathbf{x_2 x_n} \\ \vdots & \vdots & \cdots & \vdots \\ c + \mathbf{x}_m \mathbf{x_1} & c + \mathbf{x}_m \mathbf{x_2} & \cdots & c + \mathbf{x}_m \mathbf{x_n} \end{pmatrix}$$

The next step is to use property of positive semidefinite to determine c. Then through the theorem of existence of feature map, it is said that matrix $\mathbf{K}$ is PSD if and only if $K_c =< \phi(x), \phi(x) >$, for $\mathbf{x} \in \Re^n$. Hence, we have to find a corresponding feature map to prove martrix is SPD.

$$\begin{aligned} \mathbf{K} &= c + x_1 z_1 + x_2 z_2 + x_3 z_3 \dots x_n z_n \\ &= (\sqrt{c}, x_1, x_2, x_3 \dots x_n) \cdot (\sqrt{c}, z_1, z_2, z_3 \dots z_n) \\ &=< \phi(x), \phi(z) > \end{aligned}$$

Hence, we successfully defined a feature map, $\phi(x) = (\sqrt{c}, x_1, x_2, x_3 \dots x_n)$. Considering condition on $\sqrt{c}$, c has to be non-negative.
Also by property of PSD,

$$\begin{aligned} \mathbf{v}^{\mathrm{T}}(\mathbf{x}\mathbf{x}^{\mathrm{T}} + c \times A_n)\mathbf{v} &\geq 0 \\ \mathbf{v}^{\mathrm{T}}\mathbf{x}\mathbf{x}^{\mathrm{T}}\mathbf{v} + c\mathbf{v}^{\mathrm{T}} A_n \mathbf{v} &\geq 0 \\ (\mathbf{v}^{\mathrm{T}}\mathbf{x})^2 + \mathbf{v}^{\mathrm{T}}\mathbf{v} &\geq 0 \\ (\sum_{i=1}^{n} v_i x_i)^2 + c\sum_{i=1}^{n} v_i^2 &\geq 0 \end{aligned}$$

if c is non-negative, the inequality holds.
(b) In this section, we need to explore how c influences calculation.

In linear regression, we are given a set of data:

$$(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), ..., (\mathbf{x_m}, y_m)$$

where $\mathbf{x} = (x_1, ..., x_n)$ is a vector in $\Re^n$ and y is a real number. We use square loss function to find

$$w^* = \underset{w}{argmin} \sum_{i=1}^{m} (y_i - \mathbf{w} \cdot \mathbf{x_i})^2 = \underset{w}{argmin} (X\mathbf{w} - \mathbf{y})^T (X\mathbf{w} - \mathbf{y})$$

By taking derivative and make it equal 0, we can find

$$\mathbf{w} = (X^T X)^{-1} X^T \mathbf{y}$$

Slimier idea can be applied when using basis function and mapping to a higher dimensional. Taking feature map from $\phi : \Re^n \to \Re^k$

$$\phi(\mathbf{x}) = (\phi_1(\mathbf{x}), ..., \phi_k(\mathbf{x})), \mathbf{x} \in \Re^n.$$

and transforming data into,

$$((\phi_1(\mathbf{x_1}), ..., \phi_k(\mathbf{x_1})), y_1), ..., ((\phi_1(\mathbf{x_m}), ..., \phi_k(\mathbf{x_m})), y_m)$$

And use square loss to find the minimum $\mathbf{w}$, the same as before w^* can be presented as

$$w^* = (\Phi^T\Phi)^{-1}\Phi^T\mathbf{y} \tag{18}$$

For convenience let $\overrightarrow{\alpha} = (\Phi^T\Phi)^{-1}\mathbf{y}$ and $\overrightarrow{\alpha} = (\alpha_1, .., \alpha_m)^T$, then equation(3) comes out

$$w^* = \overrightarrow{\alpha}\mathbf{y} = \sum_{i=1}^{m} \alpha_i \phi(\mathbf{x_i})$$

Deduce from prime form

$$f(\mathbf{x}) = \mathbf{w^T}\Phi(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i \phi(\mathbf{x_i})^T \phi(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i K(x_i, x)$$

we get the dual form. And using feature map derived from last question to expand $f(x)$ and see what happens.

$$\begin{aligned}
\phi(x) &= (\sqrt{c}, x_1, x_2, x_3 \dots x_n) \\
f(x) &= \sum_{i=1}^{m} \alpha_i K(x_i, x) \\
&= \alpha_1(\sqrt{c}, x_{11}, x_{12}, ..., x_{1n}) \begin{pmatrix} \sqrt{c} \\ \mathbf{x_1} \\ \vdots \\ \mathbf{x_n} \end{pmatrix} \\
&+ \alpha_2(\sqrt{c}, x_{21}, x_{22}, ..., x_{2n}) \begin{pmatrix} \sqrt{c} \\ \mathbf{x_1} \\ \vdots \\ \mathbf{x_n} \end{pmatrix} \\
&+ \vdots \\
&= \alpha_m(\sqrt{c}, x_{m1}, x_{m2}, ..., x_{mn}) \begin{pmatrix} \sqrt{c} \\ \mathbf{x_1} \\ \vdots \\ \mathbf{x_n} \end{pmatrix}
\end{aligned}$$

we can plug the c out the matrix multiplication

$$\begin{aligned}
f(x) &= \alpha_1(c + x_{11}\mathbf{x_1} + ... + x_{1n}\mathbf{x_n}) \\
&\quad + \alpha_2(c + x_{21}\mathbf{x_1} + ... + x_{2n}\mathbf{x_n}) \\
&\vdots \\
&\quad + \alpha_m(c + x_{m1}\mathbf{x_1} + ... + x_{mn}\mathbf{x_n}) \\
&= c\sum_{i=1}^{m} \alpha_i + \overrightarrow{\alpha}^T X\mathbf{x}
\end{aligned}$$

where $\overrightarrow{\alpha} = (\alpha_1, .., \alpha_m)^T$, X is the $(m \times n)$ matrix containing all the value of x and $\mathbf{x} = (\mathbf{x_1}, ..., \mathbf{x_n})^T$. **Note: the x here is different from training set, this is for basis function.**Therefore, calculating the extra term $c\sum_{i=1}^{m} \alpha_i$ requires extra $O(m)$ operations. And also acting as a constant offset after mapping to the high dimensional space which help us to reduce the error caused by bias.

Question 8

Gaussian Kernel and 1-NN trained on same dataset.

Given dataset $(\mathbf{x_1}, y_1), ..., (\mathbf{x_m}, y_m) \in \Re^n \times (-1, 1)$. When using linear classification $\mathbf{y} = \mathbf{w^T x}$ is hard to predict the label of y, I decide mapping dataset to a higher dimensional space using basis function $\phi(x)$. With the same calculation in Question 7, I can get the dual form of prediction

$$f(\mathbf{t}) = \sum_{i=1}^{m} \alpha_i K_\beta(\mathbf{x_i}, \mathbf{t})$$

where $K_\beta(\mathbf{x_i}, \mathbf{t})$ is a Gaussian Kernel and $\overrightarrow{\alpha} = (\Phi^T \Phi)^{-1}\mathbf{y}$. Thus, using $sign(f(\mathbf{t}))$ as a benchmark to classify label of y.

$$f(x) = \begin{cases} 1 & sign(f(\mathbf{t})) \text{ is negative} \\ -1 & sign(f(\mathbf{t})) \text{ is positive} \end{cases} \tag{19}$$

In 1-NN algorithm, we find the nearest point of testing point x_i and classify testing point t use same label as x_i. Closeness is defined by a metric.

$$d = min\|\mathbf{x}_i - \mathbf{t}\|^2, \text{for i = 1,...,n}$$

The distance can be shown as,

$$\|\phi(\mathbf{x}) - \phi(\mathbf{t})\|^2 = < \phi(\mathbf{x}), \phi(\mathbf{x}) > + < \phi(\mathbf{t}), \phi(\mathbf{t}) > -2 < \phi(\mathbf{x}), \phi(\mathbf{t}) >$$

Euclidean gives equal weights to all the neighboring points.But one of the major drawbacks was the equal assignment of weights. To modify it with main intuition is that weights should decrease with increase in distance and more weights should be assigned for nearest points.

$$f(x) = \frac{\sum_{i=1}^{m} K_\beta(\mathbf{x_i}, \mathbf{t}) y_i}{\sum_{i=1}^{m} K_\beta(\mathbf{x_i}, \mathbf{t})}$$

Question 9

Define a $n \times n$ matrix $\mathbf{A}$ as the initial condition board configuration, a $(0, 1)$ matrix where, each 1 represents a mole appearing and 0 represents no mole appearing. The action of the hitting holes placed at (i, j) can be interpreted as the matrix addtion $\mathbf{A} + \mathbf{B}_{ij}$, where $\mathbf{B}_{ij}$ is the matrix in which the only entries equal to one are those placed at (i, j) and in the adjacent positions, there are essentially three different types of matrices $\mathbf{B}_{ij}$ depending on whether (i, j) is a corner entry.

The movement can be recorded as a polynomial in n:

$$\mathbf{A} + \sum_{i,j}^{n} \mathbf{w}_{ij} \mathbf{B}_{ij} = \mathbf{0} \tag{20}$$

where 0 denotes the zero matrix, which corresponds to the situation where all moles are hidden and each coefficient w_{ij} represents the number of times that hit (i, j) happens.