COMP0078 Supervised Learning

Course Work 1

Number: 21069508
Number: 21136404

University College London

Academic Year: 2021/2022

Contents

1 Part 1

1.1 Question 1

For each of the polynomial bases of dimension $k = 1, 2, 3, 4$ fit the data set of $(1, 3), (2, 2), (3, 0), (4, 5)$

a) First, with given k, we build the feature map $\phi(\boldsymbol{x})$ with basis $\{1, x, \cdots, x^{k-1}\}$. Then we use the expression $\boldsymbol{w} = (X^T X)^{-1} X^T \boldsymbol{y}$ to get corresponding $\boldsymbol{w}$, which is the optimal solution. Finally, we generate plenty of x to show the fitted curve with calculated $\boldsymbol{w}$.

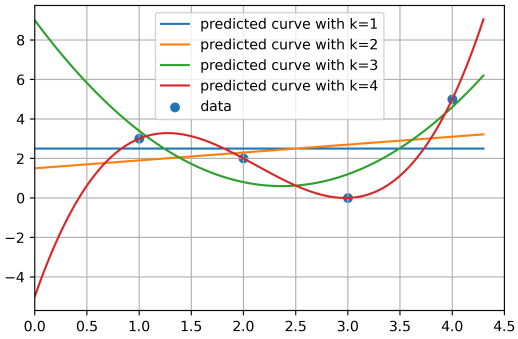

Figure 1: Different Polynomial Kernels

b) when $k = 1$, the equation is $\hat{y} = 2.5$;
when $k = 2$, the equation is $\hat{y} = 1.5 + 0.4x$;
and when $k = 3$, the equation is $\hat{y} = 9 - 7.1x + 1.5x^2$.

c) When $k = 1$, $MSE = 3.25$;
when $k = 2$, $MSE = 3.05$;
when $k = 3$, $MSE = 0.800$;
and when $k = 4$, $MSE = 5.29 * 10^{-24}$.
It can be seen from above that as k increases, the fitness of data given polynomial of x is increasing drastically.

1.2 Question 2

In this part we will illustrate the phenomena of overfitting.

a) i We first randomly generate 30 times of x from a uniform distribution with range $(0, 1)$, and we generate ϵ using a normal distribution with $mean = 0$ and $variance =$

0.07. Then we have Thus we have $g_\sigma(x) = sin^2(2\pi x) + \epsilon$, which is identically and independently distributed data. The plot of $g_\sigma(x)$ is shown as bellow.

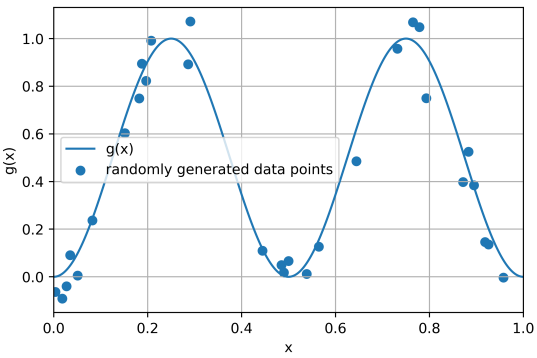

Figure 2: plot of $g_\sigma(x)$

ii We generate featured mapped data using polynomial expression to form feature mapped data matrix $\hat{\boldsymbol{X}}$. Then we use least square method to derive the closure form of $\boldsymbol{w}$, which is represented as following:

$$\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y}$$

The plots of each of these 5 curves are shown as following:

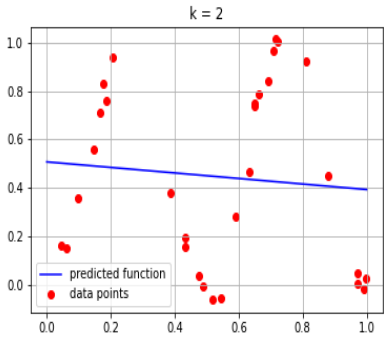

Figure 3: plot of function k=2

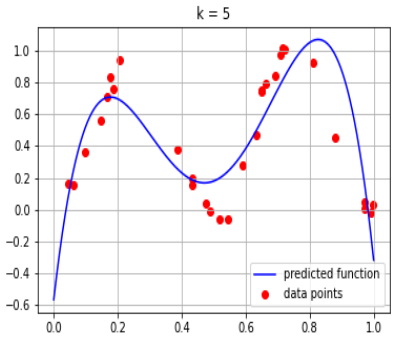

Figure 4: plot of function k=5

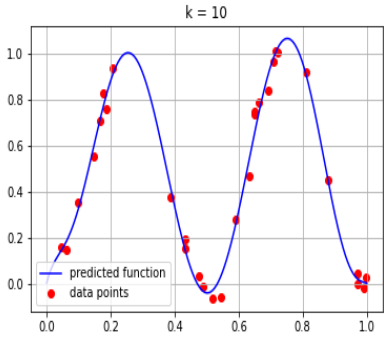

Figure 5: plot of function k=10

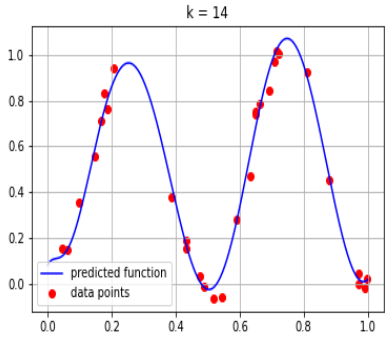

Figure 6: plot of function k=14

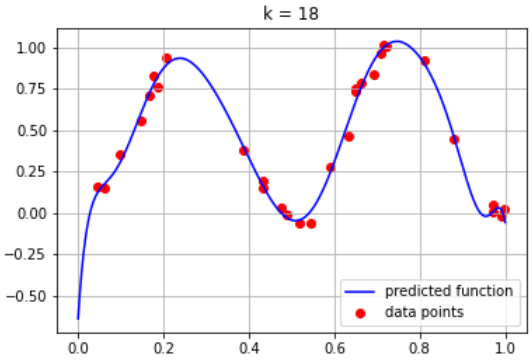

Figure 7: plot of $k = 18$

b) We plot training $\log(MSE)$ verses $k \in \{1, \cdots, 18\}$. As the plot shows, generally it is a decreasing function, while sometimes there will be an abnormal increase because of a particular data distribution.

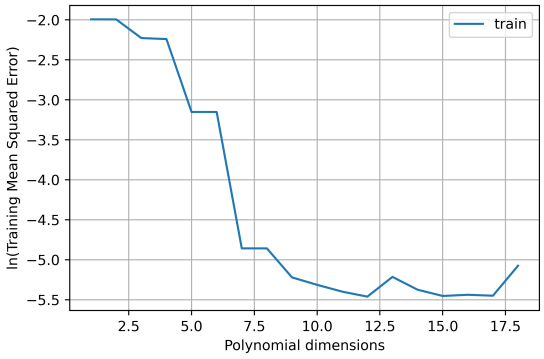

Figure 8: plot of training $\log(MSE)$

c) Then we generate 1000 test points and plot the test $\ln(MSE)$ verses $k \in \{1, \cdots, 18\}$. This time the curve is obvious not a decreasing function. When k is large, the test $\ln(MSE)$ increases because of overfitting.

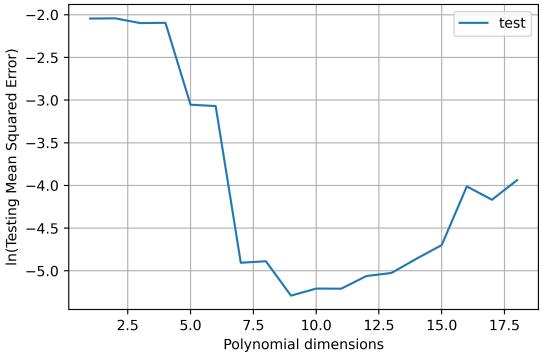

Figure 9: plot of test $\ln(MSE)$

d) Finally we run 100 epochs and then we plot the average results:

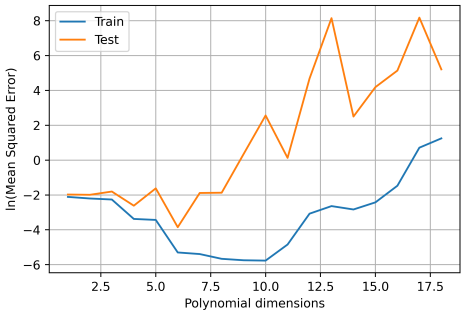

Figure 10: plot of 100 runs of train and test $\ln(MSE)$

1.3 Question 3

In this question we repeat the experiments 2(b), 2(c), and 2(d) with the same range of k but different basis $\{\sin(1\pi x), \sin(2\pi x), \cdots, \sin(k\pi x)\}$.

a) Following the instruction of 2(b), we generate 30 training points and plot the training $\log(MSE)$ verses different k's:

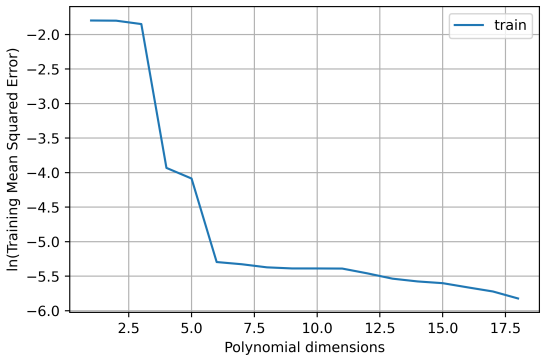

Figure 11: plot of test $\ln(MSE)$

b) Then we generate 1000 testing points and plot test $\log(MSE)$ verses different k's. It shows the overfitting when k is larger than 11:

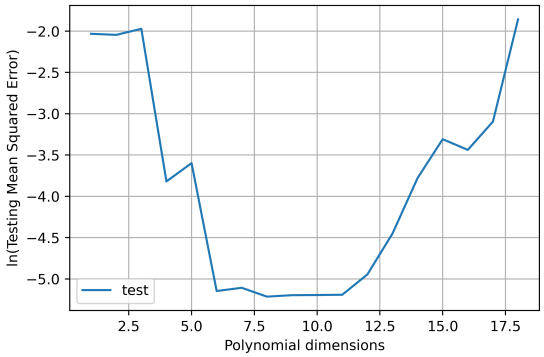

Figure 12: plot of test $\ln(MSE)$

c) Finally, we run 100 epochs repeating 3(a) and 3(b), get the average result, and plot the $\ln(avg)$. The combining curves roughly show when the model gets overfitted:

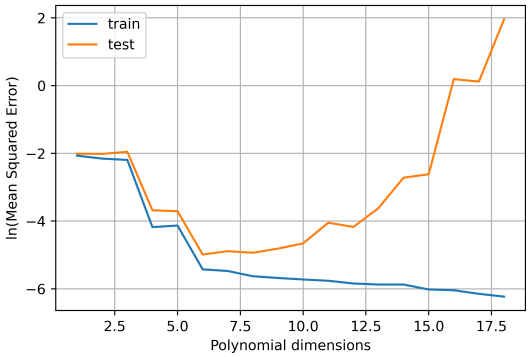

Figure 13: plot of 100 $\ln(avg)$

1.4 Question 4

In this question, we will compare the performance of Naive Regression, Linear Regression with one attribute and Linear Regression with all attributes via Mean Squared Error on both training and testing data set obtained from filtered Boston house pricing data set.

- a) **Naive Regression**
  First, we generate training data set with the vector of ones with length equal to number of training samples N_{train}, which is presented as $\mathbf{1}$ By using least square method, we can compute $\boldsymbol{w}$ by:

$$\boldsymbol{w} = (\mathbf{1}^T\mathbf{1})^{-1}\mathbf{1}^T\boldsymbol{Y}$$

  MSE of training set can be computed by:

$$\boldsymbol{MSE_{train}} = \frac{1}{\boldsymbol{N_{train}}}||\boldsymbol{w}^T\mathbf{1} - \boldsymbol{y_{train}}||^2$$

  MSE of testing set can be computed by:

$$\boldsymbol{MSE_{test}} = \frac{1}{\boldsymbol{N_{test}}}||\boldsymbol{w}^T\mathbf{1} - \boldsymbol{y_{test}}||^2$$

  after running 20 epochs on training data and testing data, we obtain the results as:

$$\boldsymbol{MSE_{train}} = \mathbf{84.20}$$
$$\boldsymbol{MSE_{test}} = \mathbf{85.11}$$

- b) **Interpretation of Naive Regression**
  after obtaining the closure form of w, we can further computed and derived as following:

$$\boldsymbol{w} = \frac{1}{\boldsymbol{N_{train}}}\sum_{i=1}^{\boldsymbol{N_{train}}}\boldsymbol{y_i}$$

The output of Naive Regression is represented as:

$$\boldsymbol{f(x) = w^T 1}$$

It can be seen that the output of Naive Regression is equal to the vector of mean of label y_i, which is the **mean value** of the housing price in the training data set.

- c) **Linear Regression with single attribute**
  In Boston housing price data set, there are 12 attributes and 1 label (housing price), we use each of 12 attributes to create our training and testing data set. For a single sample, we extract the specific attribute $\boldsymbol{x_i}$ from the original training sample and combine a addition entry "1" to incorporate bias term, so the input training data matrix $\boldsymbol{X_{train}}$ should be $\boldsymbol{[X, 1]_{N_{train}*2}}$, same as testing data.
  after obtaining the input data matrix $\boldsymbol{X_{train_j}}$ and $\boldsymbol{X_{test_j}}$ for each attribute, $j = 1, 2, ..., 12$, we can compute $\boldsymbol{w_j}$ by:

$$\boldsymbol{w_j = (X^T_{train_j} X_{train_j})^{-}1 X_{train_j} Y_{train_j}} \text{ for } j = 1, 2, ..., 12$$

  Thus, f(x) is derived as:

$$\boldsymbol{y_{train_j} = f(X_{train_j}) = X_{train_j} w_j}, \boldsymbol{y_{test_j} = f(X_{test_j}) = X_{test_j} w_j}$$

  where $y_{train_j} = f(X_{train_j})$ is the housing price in the training set predicted by the j attribute, while $y_{test_j} = f(X_{test_j}$is the housing price in the testing set predicted by the j attribute. After 20 epochs of running, the results are obtained as following:

$$\boldsymbol{MSE_{train_j} = \frac{1}{N_{train}} ||y_{train} - y_{train_j}||^2}$$

$$\boldsymbol{MSE_{test_j} = \frac{1}{N_{test}} ||y_{test} - y_{test_j}||^2}$$

  for each attribute, the training set of MSE and testing set of MSE are shown as following:

Index	Attribute	MSE train	MSE test
1	CRIM	69.36	77.80
2	ZN	71.56	77.77
3	INDUS	62.30	69.73
4	CHAS	79.85	86.46
5	NOX	66.96	73.41
6	RM	42.56	46.13
7	AGE	69.98	77.72
8	DIS	76.87	84.17
9	RAD	70.05	76.60
10	TAX	63.77	70.43
11	PTRATIO	61.11	66.36
12	LSTAT	37.13	41.58

Table 1: MSE for different training approaches

- d) **Linear Regression with all attributes**
  Different from Linear Regression with single attribute, now we create data set with all attributes and we combine an additional entry "1" to every sample to incorporate bias term. So the input training data should be $[\boldsymbol{X_{train}}, \mathbf{1}]$, same as the testing data.
  after obtaining the training data and testing data, we can compute $\boldsymbol{w}$ as following:

$$\boldsymbol{w} = (\boldsymbol{X_{train}^T X_{train}})^{-1} \boldsymbol{X_{train} Y_{train}}$$

  Thus, $f(x)$ is derived as:

$$\boldsymbol{f(X_{train})} = \boldsymbol{X_{train} w},\ \boldsymbol{f(X_{test})} = \boldsymbol{X_{test} w}$$

  where $f(X_{train})$ is the housing price in the training set predicted by the all attributes, while $f(X_{test}$is the housing price in the testing set predicted by all attributes. Finally, we compute MSE of training set and MSE of testing set as following:

$$\boldsymbol{MSE_{train}} = \frac{1}{\boldsymbol{N_{train}}} ||\boldsymbol{f(X_{train})} - \boldsymbol{y_{train}}||^2$$

$$\boldsymbol{MSE_{test}} = \frac{1}{\boldsymbol{N_{test}}} ||\boldsymbol{f(X_{test})} - \boldsymbol{y_{test}}||^2$$

  After 20 epochs of running, the results are obtained as following::

$$\boldsymbol{MSE_{train}} = \mathbf{21.35}, \boldsymbol{MSE_{test}} = \mathbf{25.90}$$

  It can be directly seen that this method outperforms any of the individual regressors that are shown above.

1.5 Question 5

In this question, we implement Gaussian Kernel on Linear Regression based on Boston housing price data set.

- a)
  We follow the procedure below to find best pair of σ^*, γ^*.

  i. We hold out $\frac{2}{3}$ of the housing price data set to obtain training set, $\frac{1}{3}$ for the testing set. Then, we split out the input and output for training and testing data set to derive $\boldsymbol{X_{train}, y_{train}, X_{test}, y_{test}}$.

  ii. We create vectors of $\boldsymbol{\sigma}$ and $\boldsymbol{\gamma}$. Then for each pair of σ_i, γ_j, we apply 5-fold cross validation to training data. For 5-fold cross validation, we split training data to approximately 5 equal folds. Then, we have 5 training runs, for each run, we choose 1 out of 5 folds to be validation set, while the rest folds form the sub-training set. In the following procedure in a), we simply regard sub-training set as $\boldsymbol{X_{train}, y_{train}}$ and validation set as $\boldsymbol{X_{val}, y_{val}}$

  iii We can derive **Gram Matrix** $\boldsymbol{G}$, where $G_{pq} = K(X_{train_p}, X_{train_q}; \sigma_i)$, for p,q in index set of sub-training set.

  iv. Then we are able to compute $\boldsymbol{\alpha} = (\boldsymbol{G} + \gamma^j * \boldsymbol{I}_{N_{train}})^{-1}\boldsymbol{y_{train}}$, where N_{train} is the number of samples in sub-training set.

  v. Once we have Gram Matrix $\boldsymbol{G}$ and $\boldsymbol{\alpha}$, for each runs on 5-fold cross validation, we can compute
  $\boldsymbol{y_{val_{pred}}} = [f(X_{val_1}); f(X_{val_2}); ...f(X_{val_{N_{val}}})]$ where $f(X_{val_j}) = \sum_{i=1}^{N_{val}} \alpha_i K(X_{train_i}, X_{val_j})$

  vi. Finally, we can derive $MSE_{val} = \frac{1}{N_{val}}||\boldsymbol{y_{val}} - \boldsymbol{y_{val_{pred}}}||^2$. We sum up the MSE_{val} for each run and divide them by 5 to obtain the final MSE_{val} for pair σ_i, γ_j.
  After 5-fold cross validation, we obtain the best pair of σ and γ by selecting the corresponding smallest cross validation error MSE_{val}. Thus, the optimal pair on one running of 5-fold cross validation is shown as:

  $$\boldsymbol{\sigma = 2^{11.5}, \gamma = 2^{-36}}$$

- b)
  The following figure presents the function of cross validation error(MSE) of σ and γ.

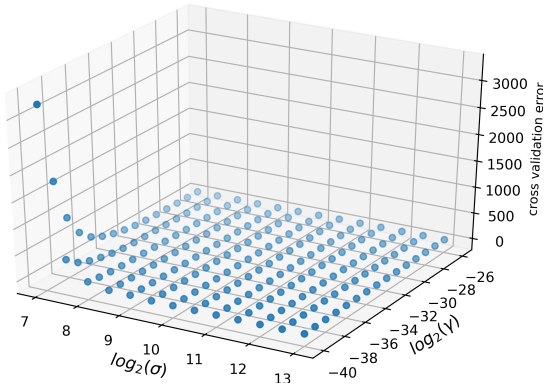

Figure 14: plot of cross validation error

- c)
  Once we obtain the optimal σ^* and γ^*, we are able to compute our MSE both on training data and testing data by following the procedure:
  i. $G_{ij} = K(X_{train_i}, X_{train_j}; \sigma^*)$

  ii. $\alpha = (G + \gamma^* * I_{N_{train}})^{-1} y_{train}$

  iii. $f(X_{train_j}) = \sum_{i=1}^{N_{train}} \alpha_i K(X_{train_i}, X_{train_j})$

  iv. $f(X_{test_j}) = \sum_{i=1}^{N_{train}} \alpha_i K(X_{train_i}, X_{test_j})$

  v. $y_{train_{pred}} = [f(X_{train_1}); f(X_{train_2}); ...f(X_{train_{N_{train}}})]$

  vi. $y_{test_{pred}} = [f(X_{test_1}); f(X_{test_2}); ...f(X_{test_{N_{test}}})]$

  vii. $MSE_{train} = \frac{1}{N_{train}} ||y_{train} - y_{train_{pred}}||^2$

  viii. $MSE_{test} = \frac{1}{N_{test}} ||y_{test} - y_{test_{pred}}||^2$

  After following the computing procedure, we obtained MSE_{train} and MSE_{test} as below:
  $$\boldsymbol{MSE_{train} = 4.97, MSE_{test} = 14.05}$$

- d) We repeat exercise 4 over 20 random splits, and put the results in the below chart:

Method	**MSE train**	**MSE test**
Naive Regression	85.13 ± 4.80	83.35 ± 9.83
Linear Regression (attribute 1)	71.60 ± 4.78	73.29 ± 10.32
Linear Regression (attribute 2)	73.91 ± 5.26	73.23 ± 10.81
Linear Regression (attribute 3)	64.79 ± 4.75	64.87 ± 9.62
Linear Regression (attribute 4)	82.57 ± 4.56	81.18 ± 9.39
Linear Regression (attribute 5)	69.30 ± 4.67	68.85 ± 9.55
Linear Regression (attribute 6)	43.56 ± 2.91	44.02 ± 5.88
Linear Regression (attribute 7)	72.77 ± 5.18	72.21 ± 10.55
Linear Regression (attribute 8)	79.72 ± 5.24	78.65 ± 10.78
Linear Regression (attribute 9)	71.95 ± 4.70	72.95 ± 9.60
Linear Regression (attribute 10)	65.62 ± 4.42	66.84 ± 8.99
Linear Regression (attribute 11)	62.41 ± 3.87	63.65 ± 7.96
Linear Regression (attribute 12)	38.42 ± 2.01	38.94 ± 4.14
Linear Regression (all attributes)	21.73 ± 1.49	24.93 ± 3.05
Kernel Ridge Regression	8.75 ± 1.26	13.37 ± 1.49

Table 2: MSE for different training approaches

2 Part 2

2.1 Question 6

First, we randomly generate 100 centres using uniform distribution on $[0,1]^2$ with 100 labels sampled uniformly from 0, 1. We use these 100 samples as training data to train a KNN model with Size = 100 and K = 3. We implement this model by computing the L2 distances between test sample and 100 train samples. Then we select the nearest 3 samples and assign the mode number among these 3 labels of these samples to the test sample. Thus we have a "data generator" $h_{S,v}$ with S =100, v = 3. We randomly generate 10000 samples using uniform distribution on $[0,1]^2$, and we use $h_{S,v}$ to derive the corresponding labels for each samples. We view the distribution of these 10000 samples by plotting them with different colors for different labels as following (Red represents label 1, while blue represents label 0):

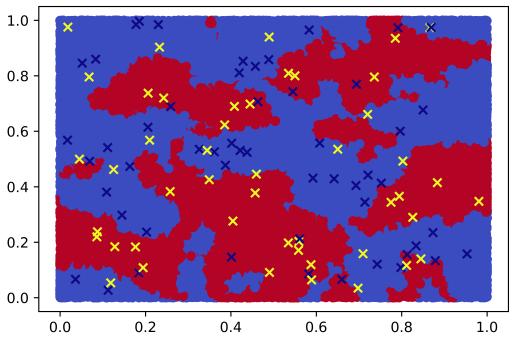

Figure 15: plot of distribution generated from $h_{100,3}$

2.2 Question 7

In protocol A, we compute generalization error by following

$$\text{generalization error} = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} I(y_{pred_i} \neq y_{test_i})$$

By implementing protocol A, we can plot the estimated generalization error versus K and it is shown as following:

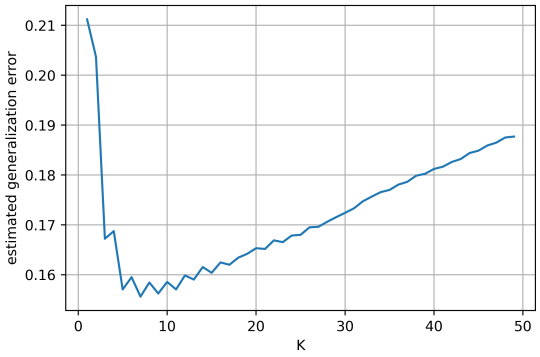

Figure 16: plot of estimated generalization error versus K on 1000 testing data samples

It can be seen that when k is in the range of (0,3), the generalization error drops from a relatively high value to the lowest point, while after that, it increases as k grows to 50. Let's set m to be the number of training samples. In this situation, m is fixed with 4000. When $\frac{m}{k}$ is large, we are tend to have too many samples while we are only focusing on local area of the test samples, therefore, we tend to easily overfitting the noise data by choosing a small area around test sample. When $\frac{m}{k}$ decreases, i.e. k increases, we will transfer from overfitting to underfitting, which means we are tend to choose the mode number in a large chosen area. Thus, the optimal solution of K lies in the range between overfitting and underfitting.

2.3 Question 8

Similar to Question 7, this time we have different number of samples to run, i.e. m $\in 100, 500, 1000, 1500, ..., 4000$, for each m, we run the protocol A to derive a generalization error and we choose K that minimize the generalization error. Thus we can plot optimal K versus m as following:

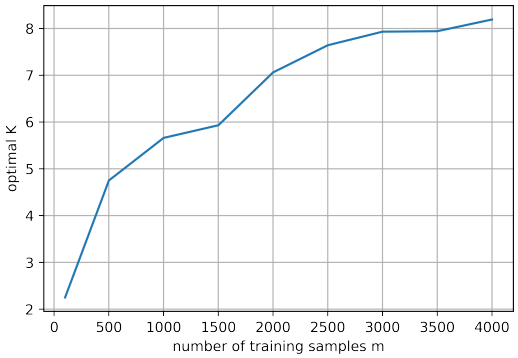

Figure 17: plot of number of training samples m versus optimal K

It can be seen from the figure that when m increases, optimal k also increases, but the increasing rate of k should be slower than that of m. As explained in [1], one can show that $\epsilon(k(m) - NN)$ will converge to $\epsilon(f^*)$ as m goes to infinity provided that:
i. $k(m)$ goes to infinity.
ii. $\frac{k(m)}{m}$ goes to 0.
where k(m) represents the k as a function of m

which means that, when we have variables m and k, the increasing rate of k should be smaller than that of m to maintain the best performance of KNN.

3 Part 3

3.1 Question 9

- a)
  Our aim is to argue that $K_c(\boldsymbol{x}, \boldsymbol{z}) = c + \sum_{i=1}^{n} x_i z_i$ is a positive semi-definite kernel. From the property of kernels, we know that **the sum of two positive semi-definite kernels is a positive semi-definite kernel**.

  Assuming $K_c(\boldsymbol{x}, \boldsymbol{z})$ is a valid kernel, then we can view $K_c(\boldsymbol{x}, \boldsymbol{z})$ as $K_1 + K_2$ where $K_1 = c$ and $K_2 = \sum_{i=1}^{n} x_i z_i$. It is common to know that the feature map of K_2 is $\phi(x) = x$ and K_2 is a positive semi-definite kernel since the kernel is directly the inner product of two vectors.

  Therefore, our aim is to prove that $K_1 = c$ is a valid positive semi-definite kernel. It is common to derive the feature map of K_1 is $\phi(x) = \sqrt{c}$. Therefore, to prove that $\phi(x) = \sqrt{c}$ exists, c has to be larger or equal to 0. First, we can prove that Then, $K_1 = c$ is symmetric, since sequence any of two input will have the same output c. Second, when we construct Gram matrix, we can prove that for any sub matrix of Gram matrix using $K_1 = c$, the sub matrix is $c * \boldsymbol{I}$,($\boldsymbol{I}$ is an identity matrix) which means that the sub matrix is a positive semi-definite matrix. Therefore, we complete the prove.

  Conclusion for $K_c(\boldsymbol{x}, \boldsymbol{z}) = c + \sum_{i=1}^{n} x_i z_i$ is a positive semi-definite kernel is shown as:

  $$\text{if and only if } c \geq 0$$

- b)
  Compared to the primal formal of linear regression, we have a bias term c containing in the inner product. Therefore, we can substitute all inner product in the linear regression with inner product + c. Then the feature map is shown as: $\phi(x) = x + \sqrt{c}$. So the whole solution process is shown as following:
  i. $G_{ij} = K(X_{train_i}, X_{train_j}; \sigma^*)$

  ii. $\alpha = (G + \gamma^* * I_{N_{train}})^{-1} y_{train}$

  iii. $f(X_{train_j}) = \sum_{i=1}^{N_{train}} \alpha_i K(X_{train_i}, X_{train_j})$

  iv. $f(X_{test_j}) = \sum_{i=1}^{N_{train}} \alpha_i K(X_{train_i}, X_{test_j})$

  v. $y_{train_{pred}} = [f(X_{train_1}); f(X_{train_2}); ...f(X_{train_{N_{train}}})]$

  where Gram matrix $\boldsymbol{G}$ is

$$\begin{pmatrix} x_1x_1 + c & x_1x_2 + c & ... & x_1x_m + c \\ x_2x_1 + c & x_2x_2 + c & ... & x_2x_m + c \\ ... & ... & ... & ... \\ x_mx_1 + c & x_mx_2 + c & ... & x_mx_m + c \end{pmatrix}$$

  Thus, when c tends to be infinity, the entry of Gram matrix will tends to be c with the influence of the inner product diminished to almost 0. Thus, $\alpha = \frac{1}{c} y_{train}$, $f(X_{test_j})$ will be computed as:

$$\begin{aligned} f(X_{test_j}) &= \sum_{i=1}^{N_{train}} \frac{1}{c} y_{train_i} (X_{train_i}^T X_{test_j} + c) \\ &= \frac{1}{c} \sum_{i=1}^{N_{train}} y_{train_i} c + \frac{1}{c} \sum_{i=1}^{N_{train}} y_{train_i} X_{train_i}^T X_{test_j} \\ &= \sum_{i=1}^{N_{train}} y_{train_i} \end{aligned}$$

  It can be seen from the above that when c tends to increase to infinity, the output of linear regression will be the summation of label in the training set.

  Thus the intuition behind this is that c influence the weight of training samples (inner product of vectors in trainning samples) when making prediction, as c grows, the influence of training samples (inner product of vectors in trainning samples) decreases, thus the prediction will be naively predicted with the summation of label of training samples.

3.2 Question 10

In this question, we want to find certain set of β s.t. ridge regression using Gaussian Kernel has the same effect of applying 1-NN method to the same data set.

Thus, we want to explicitly write out the output of the prediction of ridge regression to see the connection between ridge regression and 1-NN.

For a test sample t, we can explicitly write out the expression of $f(\boldsymbol{t}) = \sum_{i=1}^{m} \alpha_i K_\beta(\boldsymbol{x_i}, \boldsymbol{t})$ following the procedure described in previous questions:

i $G_{ij} = K_\beta(\boldsymbol{x_i}, \boldsymbol{x_j})$

ii $\boldsymbol{\alpha} = (\boldsymbol{G} + \gamma^* * \boldsymbol{I_m})^{-1}\boldsymbol{y}$

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

Notice that the diagonal element of the Gram matrix is all equal to 1 since $G_{ii} = K_\beta(\boldsymbol{x_i}, \boldsymbol{x_i}) = \exp(-\beta\|\boldsymbol{x_i} - \boldsymbol{x_i}\|^2) = 1$.
Thus we can express the Gram matrix as following according to procedure i:

$$G(\boldsymbol{x}, \boldsymbol{x}) = \begin{pmatrix} 1 & K_\beta(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & K_\beta(\boldsymbol{x}_1, \boldsymbol{x}_m) \\ K_\beta(\boldsymbol{x}_2, \boldsymbol{x}_1) & 1 & \cdots & \cdots \\ K_\beta(\boldsymbol{x}_3, \boldsymbol{x}_1) & \cdots & 1 & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ K_\beta(\boldsymbol{x}_m, \boldsymbol{x}_1) & \cdots & \cdots & 1 \end{pmatrix}$$

Note that $\boldsymbol{\alpha} = \boldsymbol{G^{-1}y}$, in $\boldsymbol{\alpha}$, there is an inverse matrix that we cannot explicitly express, so we use a_{ij} to express each entry of the matrix. And we denote its inverse matrix as:

$$G(\boldsymbol{x}, \boldsymbol{x})^{-1} = \begin{pmatrix} 1 & a_{12} & \cdots & a_{1m} \\ a_{21} & 1 & \cdots & \cdots \\ a_{31} & a_{32} & 1 & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & \cdots & \cdots & 1 \end{pmatrix}$$

Therefore, we can explicitly write out as following for each entry of $\boldsymbol{\alpha}$ according to procedure ii:

$$\begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \cdots \\ \alpha_m \end{pmatrix} = \begin{pmatrix} 1 & a_{12} & \cdots & a_{1m} \\ a_{21} & 1 & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & \cdots & \cdots & 1 \end{pmatrix} \cdot \begin{pmatrix} y_1 \\ y_2 \\ \cdots \\ y_m \end{pmatrix} = \begin{pmatrix} y_1 + y_2 a_{12} + \cdots + y_m a_{1m} \\ y_1 a_{21} + y_2 + \cdots + y_m a_{2m} \\ \cdots \\ y_1 a_{m1} + y_2 a_{m2} + \cdots + y_m \end{pmatrix}$$

After that, according to procedure iii, we can write out the prediction of test sample $\boldsymbol{t}$ as following:

$$\begin{aligned} f(t) = &(y_1 + y_2 a_{12} + \cdots + y_m a_{1m}) \cdot K_\beta(\boldsymbol{x}_1, \boldsymbol{t}) \\ &+ (y_1 a_{21} + y_2 a_{22} + \cdots + y_2) \cdot K_\beta(\boldsymbol{x}_2, \boldsymbol{t}) \\ &+ \cdots \\ &+ (y_1 a_{m1} + y_2 a_{m2} + \cdots + y_m) \cdot K_\beta(\boldsymbol{x}_m, \boldsymbol{t}) \end{aligned}$$

we re-write as:

$$\begin{aligned} f(t) = &y_1(K_\beta(\boldsymbol{x}_1, \boldsymbol{t}) + a_{21} K_\beta(\boldsymbol{x}_2, \boldsymbol{t}) + \cdots + a_{m1} K_\beta(\boldsymbol{x}_m, \boldsymbol{t})) \\ &+ y_2(K_\beta(\boldsymbol{x}_2, \boldsymbol{t}) + a_{12} K_\beta(\boldsymbol{x}_1, \boldsymbol{t}) + \cdots + a_{m2} K_\beta(\boldsymbol{x}_m, \boldsymbol{t})) \\ &+ \cdots \\ &+ y_m(K_\beta(\boldsymbol{x}_m, \boldsymbol{t}) + a_{1m} K_\beta(\boldsymbol{x}_1, \boldsymbol{t}) + \cdots + a_{m-1m} K_\beta(\boldsymbol{x}_{m-1}, \boldsymbol{t})) \end{aligned}$$

If we define $\boldsymbol{x_p}$ is the nearest training sample point to the test sample point $\boldsymbol{t}$, then we can easily write out $f(\boldsymbol{t})$ as:

$$\begin{aligned} f(t) = &y_1(K_\beta(\boldsymbol{x}_1, \boldsymbol{t}) + a_{21} K_\beta(\boldsymbol{x}_2, \boldsymbol{t}) + \cdots + a_{m1} K_\beta(\boldsymbol{x}_m, \boldsymbol{t})) \\ &+ y_2(K_\beta(\boldsymbol{x}_2, \boldsymbol{t}) + a_{12} K_\beta(\boldsymbol{x}_1, \boldsymbol{t}) + \cdots + a_{m2} K_\beta(\boldsymbol{x}_m, \boldsymbol{t})) \\ &+ \cdots \\ &+ y_p(K_\beta(\boldsymbol{x}_p, \boldsymbol{t}) + a_{1p} K_\beta(\boldsymbol{x}_1, \boldsymbol{t}) + \cdots + a_{mp} K_\beta(\boldsymbol{x}_m, \boldsymbol{t})) \\ &+ \cdots \\ &+ y_m(K_\beta(\boldsymbol{x}_m, \boldsymbol{t}) + a_{1m} K_\beta(\boldsymbol{x}_1, \boldsymbol{t}) + \cdots + a_{m-1m} K_\beta(\boldsymbol{x}_{m-1}, \boldsymbol{t}) \end{aligned}$$

We observe that the prediction of this test sample point is the weighted sum of the labels in the training data set. We can regard it as:

$$f(\boldsymbol{t}) = (K_\beta(\boldsymbol{x}_1, \boldsymbol{t}) + w_1) y_1 + \cdots + (K_\beta(\boldsymbol{x}_p, \boldsymbol{t}) + w_p) y_p + \cdots + (K_\beta(\boldsymbol{x}_m, \boldsymbol{t}) + w_m) y_m$$

Lemma 1: Since $\boldsymbol{x}_p$ is the nearest point to $\boldsymbol{t}$, only if $\beta > 0$ can we achieve our goal to simulate 1-NN.

Proof of Lemma 1:

We want the label of $\boldsymbol{x}_p$ is the output of the prediction $f(\boldsymbol{t})$.

Since $\boldsymbol{y} \in \{-1, 1\}$, we want the weight of y_p becomes the largest among all training samples, i.e. $K_\beta(\boldsymbol{x}_p, \boldsymbol{t}) = \max\{K_\beta(\boldsymbol{x}_1, \boldsymbol{t}), \cdots, K_\beta(\boldsymbol{x}_m, \boldsymbol{t})\}$.

As $\boldsymbol{x}_p$ is the nearest point to $\boldsymbol{t}$, we have the inequality: $\|\boldsymbol{x}_p - \boldsymbol{t}\|^2 < \|\boldsymbol{x}_i - \boldsymbol{t}\|^2, \forall i \in \{1, \cdots, m\}$.

1) Suppose $\beta < 0$:

   Then $K_\beta(\boldsymbol{x}_i, \boldsymbol{t}) = \exp(-\beta \|\boldsymbol{x}_i - \boldsymbol{t}\|^2)$ is a strictly increasing function with respect to

$\boldsymbol{x}_i, \forall i \in \{1, \cdots, m\}$.
So $K_\beta(\boldsymbol{x}_p, \boldsymbol{t})$ is the minimum, contradicting to our goal.

2) Suppose $\beta = 0$:
Then $K_\beta(\boldsymbol{x}_i, \boldsymbol{t}) = 1, \forall i \in \{1, \cdots, m\}$, contradicting to our goal.

3) Suppose $\beta > 0$:
Then $K_\beta(\boldsymbol{x}_i, \boldsymbol{t}) = \exp(-\beta \|\boldsymbol{x}_i - \boldsymbol{t}\|^2)$ is a strictly decreasing function with respect to $\boldsymbol{x}_i, \forall i \in \{1, \cdots, m\}$.
So $K_\beta(\boldsymbol{x}_p, \boldsymbol{t})$ is the maximum, which is our goal.

Lemma 2: Certain β should be found s.t. $K_\beta(\boldsymbol{x}_p, \boldsymbol{t}) + w_p > \sum_{i \in \{1, \cdots, m\} \setminus \{p\}} |K_\beta(\boldsymbol{x}_i, \boldsymbol{t}) + w_i|$.

Proof of Lemma 2:
Since $\boldsymbol{y} \in \{-1, 1\}$, the extreme situation is that $y_p = 1$ and $y_i = -1, \forall i \in \{1, \cdots, m\} \setminus \{p\}$. Therefore, only if we have Lemma 2 can we guarantee that $f(\boldsymbol{t})$ will be predict as y_p.

Note that Lemma 2 is the sub-set of Lemma 1 for valid β.

In conclusion, only when we have the Lemma 2:
Certain β should be found s.t. $K_\beta(\boldsymbol{x}_p, \boldsymbol{t}) + w_p > \sum_{i=1}^{m/p} |K_\beta(\boldsymbol{x}_i, \boldsymbol{t}) + w_i|$
can we guarantee that the ridge regression with Guassian kernel will have the same effect as 1-NN method, where $\boldsymbol{x}_p$ is the nearest point of test sample $\boldsymbol{t}$; $w_i = \sum_{r \in \{1, \cdots, m\} \setminus \{i\}} a_{ri} K_\beta(\boldsymbol{x}_r, \boldsymbol{t})$; a_{ri} is the r-th row, i-th column of the inverse of the Gram matrix.

3.3 Question 11

From the description of question 11, we observe that the change of the state of the hole is only related to the holes that surround it. For example, as the following picture shows:

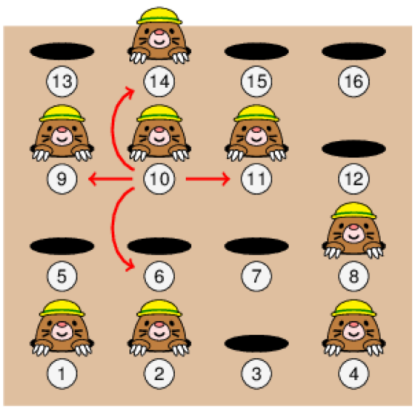

Figure 18: plot of moles

The change of the state of hole 10 is only related with the change of the state of hole 9, hole 6, hole 14, hole 11 and hole 10. Here is the explanation.

If we define a hit on i-th hole as action i ($A_i = -1$). The state of a hole (X_i) is 1 if there is **no** mole in the hole. The state of a hole is -1 if there is a mole in the hole.

Observation: The change of the state of i-th hole is only related to the holes surrounded it but has nothing to do with the sequence of change of holes surrounded it.

Let's focus on the state of hole 10. Now, if we hit hole 9, 14, 11, 6 and 10, then $X'_{10} = A_9 * A_{14} * A_{11} * A_6 * A_{10} * X_{10} = -1 * -1 * -1 * -1 * -1 * -1 = 1 = X_{10} * A_{10} * A_6 * A_{11} * A_{14} * A_9 * = -1 * -1 * -1 * -1 * -1 * -1$, which release further observation that X'_{10} is not related to the sequence of the hit around it, but only related with the hit itself. Since hit action is only "-1" with no information of the sequence.

Sub-goal: We want the state of i-th hole is equivalent to the product of the change on this hole.

For example, $X_{10} = -1$, $X'_{10} = A_9 * A_{14} * A_{11} * A_6 * A_{10} * X_{10}$, our goal is to make $X'_{10} = 1$, which means we would like $A_9 * A_{14} * A_{11} * A_6 * A_{10} = X_{10}$. The same situation happens if $X_{10} = 1$.

After obtaining observation and sub-goal, we have our final goal: $\boldsymbol{[X_1; X_2; ...; X_{n^2}] = [1, 1, ...1]^T}$

Let's now consider a toy case, with n = 3 hole matrix ($\boldsymbol{X}$).

$$\begin{pmatrix} X_7 & X_8 & X_9 \\ X_4 & X_5 & X_6 \\ X_1 & X_2 & X_3 \end{pmatrix}$$

From the observation, we can conclude that:

$$\begin{aligned} X_1 &= A_1 A_2 A_4 \\ X_2 &= A_1 A_2 A_3 A_5 \\ X_3 &= A_2 A_3 A_6 \\ X_4 &= A_1 A_4 A_5 A_7 \\ X_5 &= A_2 A_4 A_5 A_6 A_8 \\ X_6 &= A_5 A_6 A_9 \\ X_7 &= A_4 A_7 A_8 \\ X_8 &= A_7 A_8 A_9 A_5 \\ X_9 &= A_6 A_8 A_9 \end{aligned}$$

Further, we observe that the states of these 9 holes is the product of specific actions. Therefore, if we consider the product as summation and we create a vector of action with size n^2, then we can use the matrix multiplication of state transition matrix($\boldsymbol{T}$) and action vector ($\boldsymbol{A}$) to express above states.

Thus, we convert our initial define of the state and action of a hole. We define the state of i-th hole $X_i = 1$ if there **is** a mole in the hole and $X_i = 0$ vice versa. If there is action on i-th hole, then $A_i = 1$. For example, if we hit hole 9, 14, 11, 6 and 10, then $X'_{10} = X_{10} + A_9 + A_{14} + A_{11} + A_6 + A_{10} = 1 + 1 + 1 + 1 + 1 + 1 = 6$. Then we take the remainder of X'_{10} **divided by 2**, which is 0.
Thus our observation transfers to the sum of action around i-th hole is equal to the state of the i-th hole.

Our goal transfers to TA = X, with $\boldsymbol{X} = [X_1, X_2, ...X_9]^T$, note that we should take the remainder of each entry in the result of TA divided by 2.

Thus the transition matrix ($\boldsymbol{T}$) is shown as below:

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

the action vector ($\boldsymbol{A}$) is shown as below:

$$\begin{pmatrix} A1 \\ A2 \\ A3 \\ A4 \\ A5 \\ A6 \\ A7 \\ A8 \\ A9 \end{pmatrix}$$

Thus, we have expression $TA = X$, with $\boldsymbol{X} = [X_1, X_2, ...X_9]^T$, note that we should take the remainder of each entry in the result of TA divided by 2. Our aim is to solve this linear equation and this is solid for any n >0. Thus, If there is a solid solution, then we can have a sequence of hit, so that there is no mole showing up in the hole, otherwise, there is no solution for the specific $\boldsymbol{X}$ matrix.

References

[1] T. Cover and P. Hart. "Nearest neighbor pattern classification". In: *IEEE Transactions on Information Theory* 13.1 (1967), pp. 21–27. DOI: 10.1109/TIT.1967.1053964.