COMP0086 Summative Assignment

Alexei Pisacane and Jian Shu (James) Wu

Nov 16, 2022

1 PART I

1.1 Linear Regression

1. (a) Superimposing the four different curves corresponding to each fit over the four given data points:

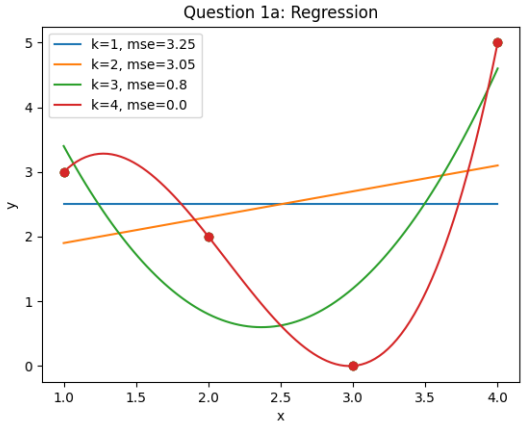

Figure 1: Data set $\{(1,3),(2,2),(3,0),(4,5)\}$ fitted with basis $\{1,x\}$ and basis $\{1,x,x^2,x^3\}$

(b) The weights for the equations:

Basis	**k=1**	**k=2**	**k=3**	**k=4**
1	0.0	0.0	0.0	-5.0
x^1	0.0	0.0	9.0	15.17
x^2	0.0	1.5	-7.1	-8.5
x^3	2.5	0.4	1.5	1.33

(c) The mean squared error for each fitted curve:

Metric	**k=1**	**k=2**	**k=3**	**k=4**
MSE	3.25	3.05	0.8	0.0

2. (a) i. Superimposing the function $sin^2(2\pi x)$ in the range $0 \leq x \leq 1$ with the points of the above data set superimposed

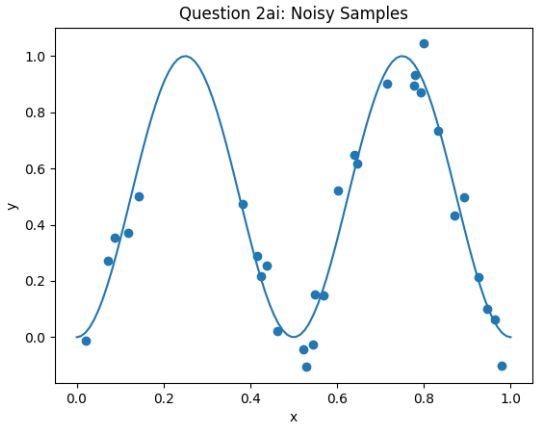

Figure 2: Noisy samples

ii. Fitting the data set with a polynomial bases of dimension $k = 2, 5, 10, 14, 18$ and plotting each of these 5 curves superimposed over a plot of data points:

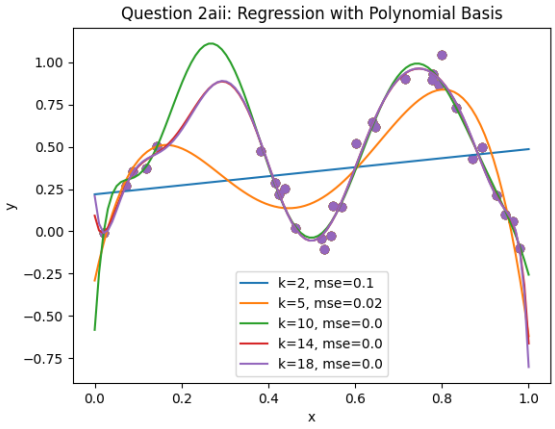

Figure 3: Polynomial fit curves

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

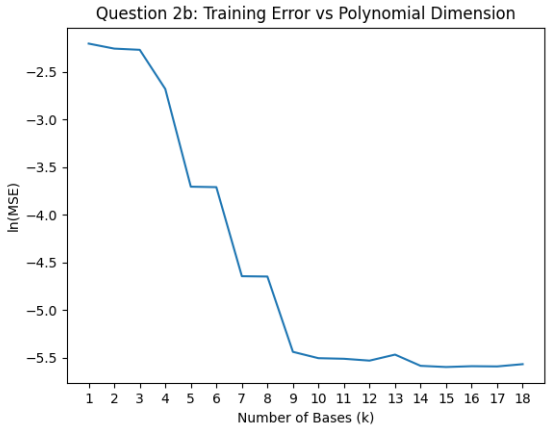

Figure 4: Training Error of Polynomial Curves

(c) Plotting the ln of the test error for a single trial of 1000 test points versus the polynomial dimension $k = 1, ..., 18$:

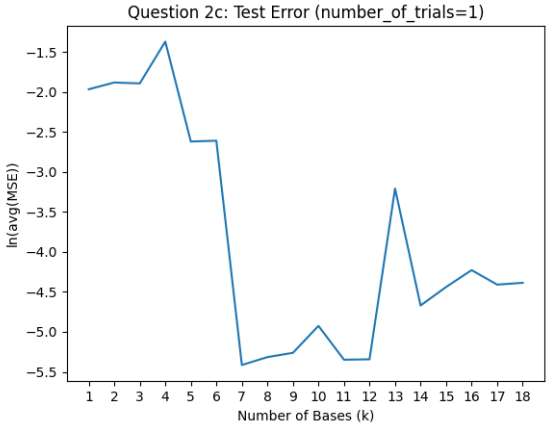

Figure 5: Testing Error of Polynomial Curves (1 trial)

(d) Plotting the ln of the test error for 100 trials of 1000 test points versus the polynomial dimension $k = 1, ..., 18$:

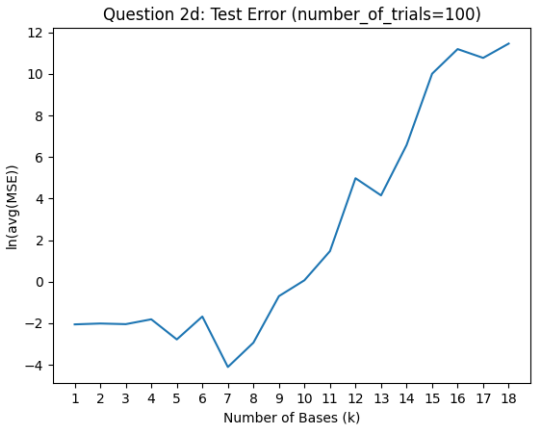

Figure 6: Testing Error of Polynomial Curves (100 trials)

3. Repeating 2 (b-d) with sin basis functions:

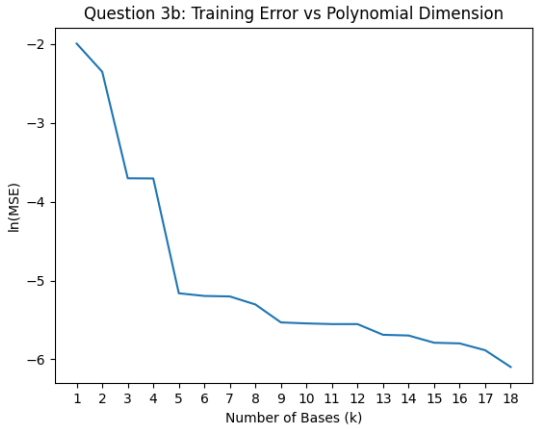

Figure 7: Training Error of sin Curves

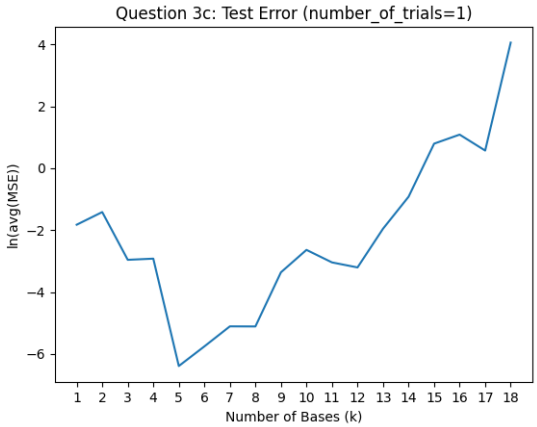

Figure 8: Testing Error of sin Curves (1 trial)

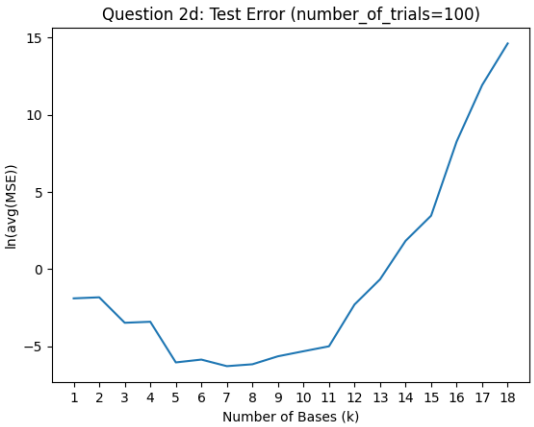

Figure 9: Testing Error of sin Curves (100 trials)

1.2 Filtered Boston housing and kernels

4.

1.3 Kernelised ridge regression

5.

2 PART II

2.1 k-Nearest Neighbors

6. Generating a random h:

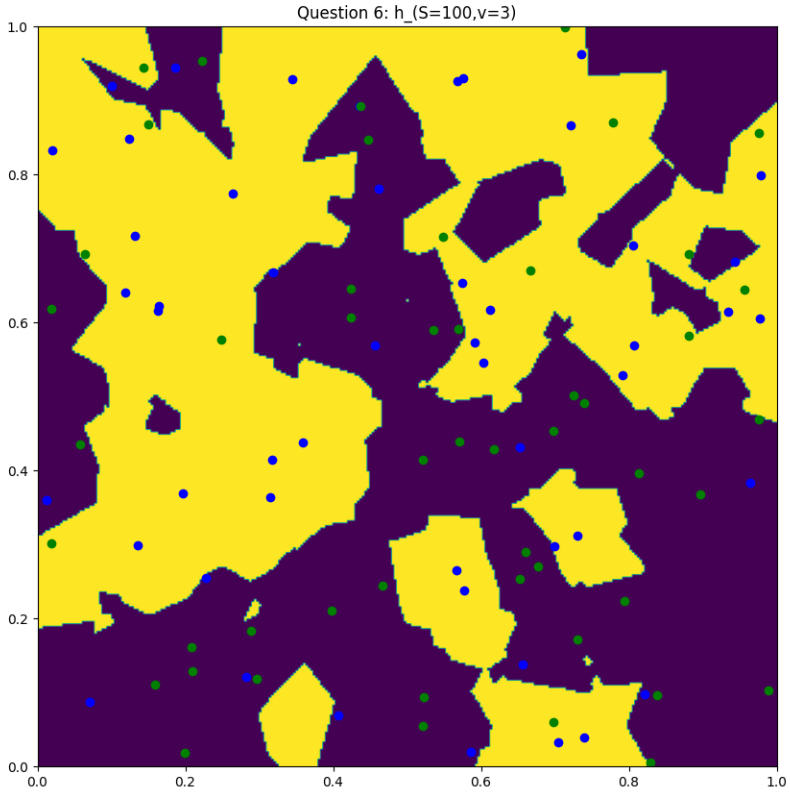

Figure 10: A hypothesis $h_{S,v}$ visualised with $|S| = 100$ and $v = 3$

7. (a) Producing a visualisation using Protocol A:

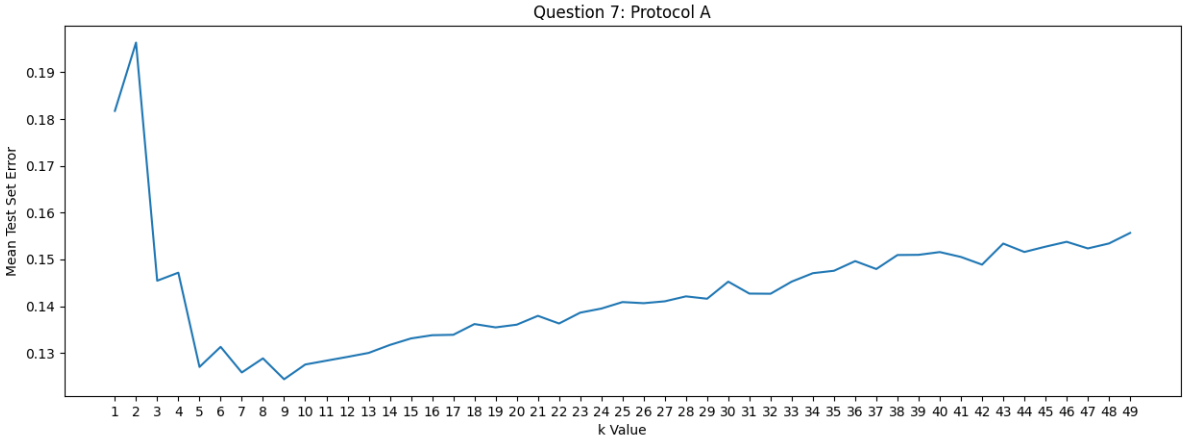

Figure 11: Protocol A

(b) We know that the data generating process uses $v = 3$, so we would have initially expected the smallest test error to be located at the matching k-value, $k = 3$. However, the data generating process for the training data set labels *and* the testing data set labels were injected with noise, so k-values greater than 3 have a "de-noising" effect allowing for better performance at around $k = 9$. As the k-value increases beyond $k = 9$, this smoothing effect is too strong, causing the gradual increase in test error as the k value increases. For $k < 9$ and in particular $k < v = 3$ we see the error growing because the model has small k and is able to fit to the noise injected into the data generating process, corrupting the model. By averaging over 100 runs, the plot is relatively smooth allowing us to make generalisations about the relationship between k-value and mean test set error.

8. (a) Producing a visualisation using Protocol B:

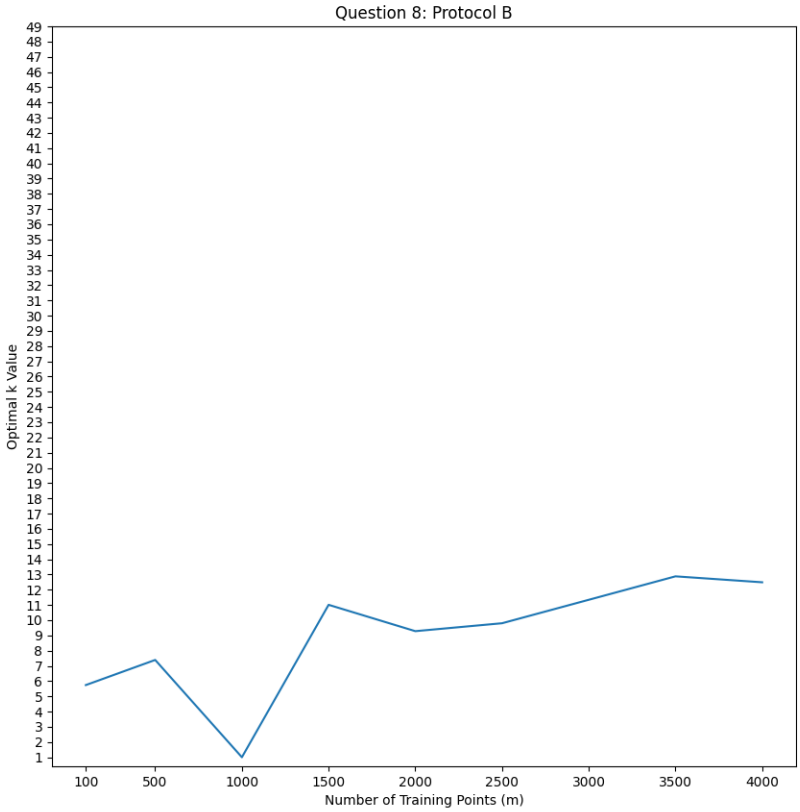

Figure 12: Protocol B

(b) TODO

3 PART III

3.1 Questions

9. (a) let $K_c(\mathbf{x}, \mathbf{z}) = c + \sum_{i=1}^{n} x_i z_i = c + \mathbf{x}^T \mathbf{z}$
Our function $K_c(\cdot, \cdot)$ is psd iff:
$\forall \{u_i\}_{1:m} \in \mathcal{R}, \{\mathbf{x}_j\}_{1:m} \in \mathcal{R}^n$:
$\sum_{k,l} u_k u_l K_c(\mathbf{x}_u, \mathbf{x}_l) \geq 0$ We expand our kernel function to give:

$$\sum_{k,l} u_k u_l K_c(\mathbf{x}_u, \mathbf{x}_l) = \sum_{n,m} c u_n u_m + \sum_{n,m} \langle \mathbf{x}_n, \mathbf{x}_m \rangle u_n u_m$$

Here, we note that $\langle \cdot, \cdot \rangle$ is an inner product over the Hilbert space $\mathcal{R}^n$ and hence is psd by the representer theorem:

$$\implies \sum_{n,m} \langle \mathbf{x}_n, \mathbf{x}_m \rangle u_n u_m \geq 0 \tag{1}$$

Proposition
$K_c(\cdot, \cdot)$ is psd iff $c \geq 0$
Proof:
($\implies$)

Suppose $c \geq 0$.
(1) $\implies \sum_{k,l} u_k u_l K_c(\mathbf{x}_u, \mathbf{x}_l) \geq c \sum_{n,m} u_n u_m = c(\sum_n u_n)^2 \geq 0$

($\impliedby$)

let $c < 0$. Suppose our vectors x_n are identically equal to $(\sqrt{a}, 0)^T$, for some $a < |c|/n \implies x_k^T x_l = a$
$\forall k, l$.

$\implies \sum_{k,l} u_k u_l K_c(\mathbf{x}_u, \mathbf{x}_l) = (a + c) \sum_{n,m} u_n u_m = (a + c)(\sum_n u_n)^2 < 0$
Hence for any $c < 0$, $\exists \{u_i\}_{1:m} \in \mathcal{R}, \{\mathbf{x}_j\}_{1:m} \in \mathcal{R}^n$s.t:
$\sum_{k,l} u_k u_l K_c(\mathbf{x}_u, \mathbf{x}_l) < 0 \square$

(b) Using this kernel in our ridge regression, we arrive at the following prediction function:

$$\hat{y}_{test} = lc + \sum_i \alpha_i x_i \cdot x_{test}$$

where $\alpha = (K + \gamma l I_l + c 1_{nxn})^{-1} y_{train}$
Where 1_{nxn} is a matrix of all ones.

We note that as c becomes large, $\alpha \to 0$ and hence the predictions for $\hat{y}_{test}$ approach the constant function $f(x) = lc$

10. Define $g(t)$:=sgn$(f(t)) = \frac{f(t)}{|f(t)|}$
Proposition:

as $\beta \to \infty$, $g(t) \to 1 - NN$

Proof:

$f(t) = \sum_i \alpha_i K(x_i, t)$, where $\alpha = (K + \gamma I_l)^{-1} y$

since $K(x, x) = exp(0) = 1$ for all x, we decompose K as follows:

$K = I_l + \tilde{K}$, where $\tilde{K}_{ij} = K(x_i, x_j)$, $i \neq j$. We note here that as $\beta \to \infty$, $\tilde{K} \to 0$.

We take a taylor expansion in $\tilde{K}$, to arrive at the following result:

$\alpha = ((\gamma + 1)I_l + \tilde{K})^{-1} y = \frac{1}{\gamma+1}(I_l - O(\tilde{K}))y$ Hence as $\beta \to \infty$, $\alpha \to \frac{1}{\gamma+1} y$
Hence our predictor function becomes:
$f(t) \to \frac{1}{\gamma+1} \sum_i y_i K(x_i, t) \implies g(t) \to \frac{\sum_i y_i K(x_i,t)}{|\sum_i y_i K(x_i,t)|}$

Let's assume that there exists a unique point x in our training set of minimal distance to our test point.

Define $d_i := \|x_{test} - x_i\|^2$, and $d* := min_i d_i$

$g(t) = \frac{y_* exp(-\beta d_*)}{|\sum_i y_i exp(-\beta d_i)|} + \sum_{i \neq *} \frac{y_i exp(-\beta d_i)}{|\sum_i y_i exp(-\beta d_i)|} = \frac{y_*}{|y_* + \sum_{i \neq *} y_i exp(-\beta(d_i - d_*))|} + \sum_{i \neq *} \frac{y_i exp(-\beta(d_i - d_*))}{|y_* + \sum_{i \neq *} y_i exp(-\beta(d_i - d_*))|}$.

Note for $d_i \neq d_*$, $exp(-\beta(d_i - d_*)) \to 0$ as $\beta \to \infty$. Hence:
$g(t) \to \frac{y_*}{|y_* + 0|} + \frac{0}{|y_* + 0|} = y_* = y_i : x_i = min_i \|x_{test} - x_i\|$ which is the required 1-NN predictor.
□

11. Given a whack-a-mole grid of size n, it is solvable if we can apply a sequence from our n^2 permissible moves that results in a blank screen. We first index all permissible moves by $T_{i,j}$, which has the effect of flipping the state of squares $(i, j), (i-1, j), (i+1, j), (i, j-1)(i, j+1)$, as long as each coordinate is still on the grid.
We may consider each grid as an element of the space of square matrices over $\mathcal{F}_2$, the finite field of order 2: under this construction, we see that the application of a move $T_{i,j}$ corresponds to the sum between the grid and a matrix T_{ij}, where T_{ij}, where the entries of T_{ij} are zero everywhere except at $(i, j), (i-1, j), (i+1, j), (i, j-1)(i, j+1)$, if these values are present on the grid. For example, if n = 4, the transformation $T_{2,2}$ corresponds to the addition of the matrix T_{22}:

$$T_{22} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Since our operations on a grid are now simply the sum of matrices over a $\mathcal{F}_2$, it is now immediately obvious that our operations are commutative, by the commutativity of addition.

Further, since applying the same operation twice would be the same as applying the matrix twice in order, we can see that any operation applied twice would have the effect of applying no operation at all (since $T_{ij} + T_{ij} = 0$).
Hence, we conclude for $1 \leq i, j \leq n$, each operation is applied at most once. From this point, we note that there is no reason for our operations to be square matrices, as the only required properties of the space are those of a vector space (we make no use here of matrix multiplication). Hence, we may vectorise both our input grid and our operation matrices to get a set of vectors over $F_2^{n^2}$. We now index our operation vectors $t_i : i = 1, \ldots, n^2$.
Let $x_i = \mathcal{I}[t_i \text{ is used }]$ be an indicator function representing whether t_i is added to the input to get our blank grid.
We observe that our problem is solvable for a given input grid G (now vectorised as a vector $\mathbf{g}$) iff there exists a vector $\mathbf{x} \in \mathrm{F}_2^{n^2}$ st:
$\sum_i t_i x_i + \mathbf{g} = \mathbf{0}$.
If we stack the vectors t_i horizontally as the columns of a matrix T, we get:
$T\mathbf{x} + \mathbf{g} = 0$
Since every element is its own additive inverse over $\mathcal{F}_2$, we add $\mathbf{g}$ to both sides and arrive at the linear system $T\mathbf{x} = \mathbf{g}$.
Since the matrix T may not be invertible in general, we perform gaussian elimination over $\mathcal{F}_2$, i.e create an augmented matrix $[T|\mathbf{g}]$ and reduce to row echelon form. If the problem is solvable, we will end up with a consistent (but perhaps underdetermined) reduced system of equations. If the input has no solution, the reduced system of equations will have some inconsistency in its variables: there will be some row of the reduced row echelon augmented matrix with all zeros to the left of the augmentation line, and a non-zero entry to the right of the augmentation line.

Finally, we make a note of the time complexity of this algorithm: the creation of our matrix T can be performed in $O(n^4)$ since we have n^2 operation grids of size n^2. Further, gaussian elimination can be performed in $O(n^3)$ operations in any field, and hence our total running time is $O(n^3)$.

Once we have determined whether a grid configuration is solvable, it remains to find a valid solution.

We note that if a solution exists, then a least squares solver should find it. This operation should take $O(n^3)$ operations.