COMP0078: Supervised Learning **2024/25**

Coursework - ⟨1⟩

Due Date: ⟨**Nov, 20**⟩ *Student(s):* ⟨⟩

Instructions This is a sample template to submit coursework for COMP0078. It is not mandatory to use this template for submissions. This is provided just as a reference.

1.1 Linear Regression

Question 1

(a) The graph is shown in below:

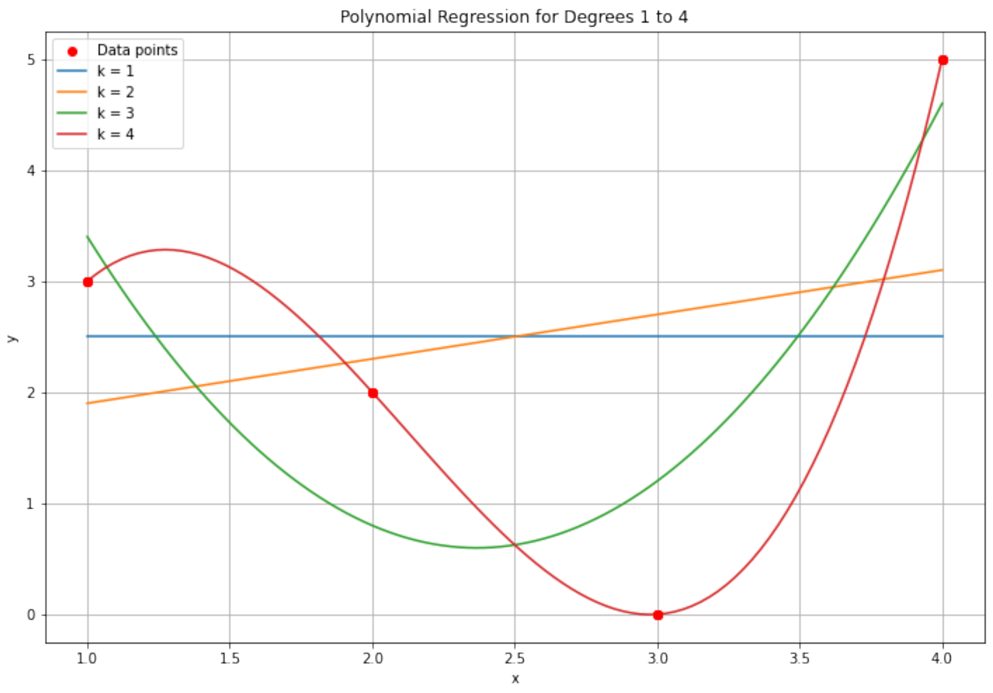

Figure 1.1. Fitted Curves for k = 1,2,3,4

(b) The polynomial equations corresponding to the curves fitted for k= 1,2,3 are as follows:

For $k = 1$:

$$y = 2.5$$

For $k = 2$:

$$y = 1.50 + 0.40x$$

For $k = 3$:

$$y = 9.00 - 7.10x + 1.50x^2$$

For $k = 4$:

$$y = -5.00 + 15.17x - 8.50x^2 + 1.33x^3$$

(c) For each fitted curve k= 1,2,3,4, the mean square error are as follows:

For $k = 1$:

$$MSE = 3.25$$

For $k = 2$:

$$MSE = 3.05$$

For $k = 3$:

$$MSE = 0.8$$

For $k = 4$:

$$MSE = 2.897717083349825 \times 10^{-23}$$

Question 2

(a) **i** The function is shown in below:

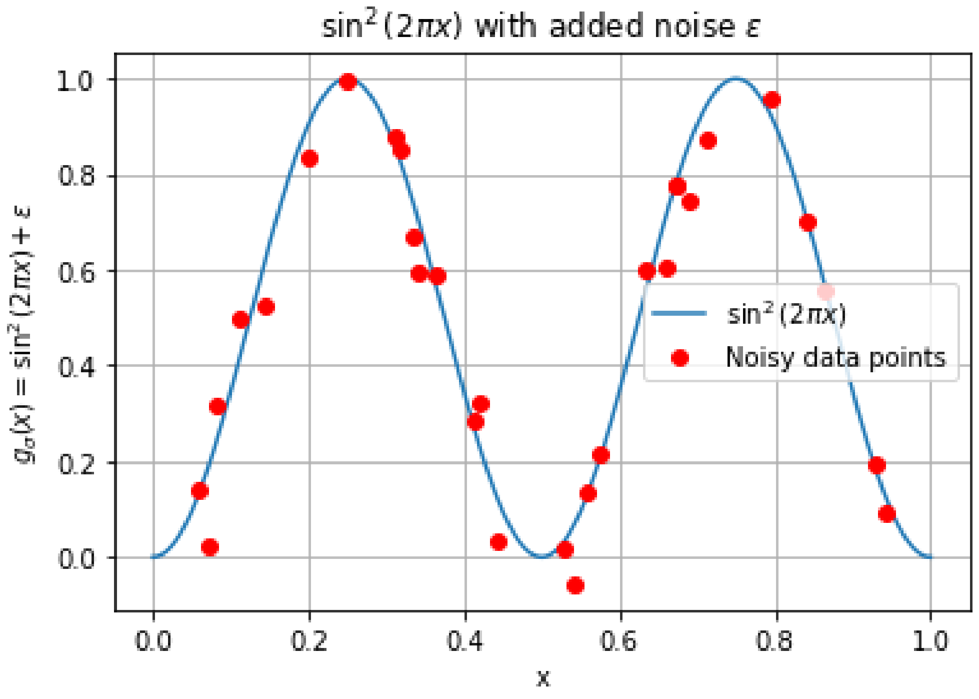

Figure 1.2. The function $\sin^2(2\pi x)$ in $[0, 1]$

ii The graph is shown in below:

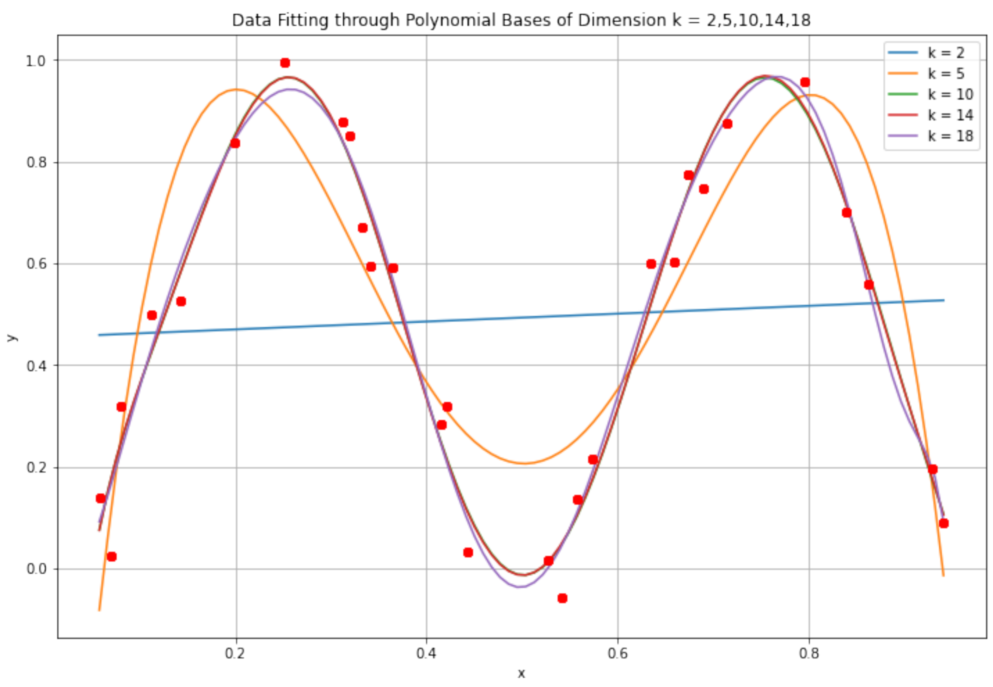

Figure 1.3. Fitted Curves for k = 2,5,10,14,18

(b) See figure 1.4.

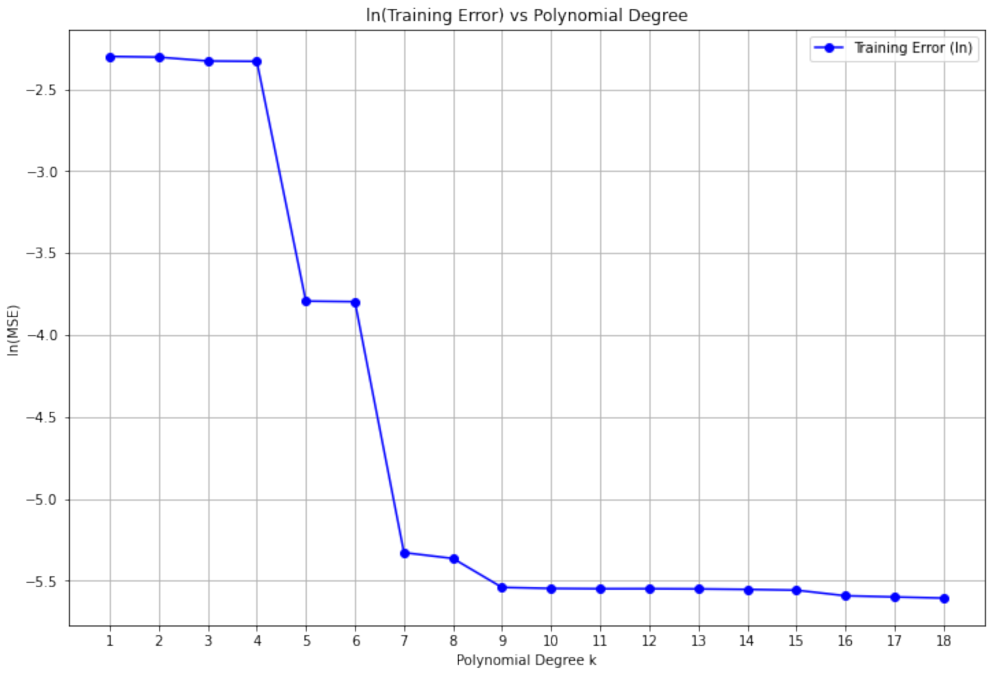

Figure 1.4. ln(training error) versus the polynomial dimension k= 1,...,18

(c) See figure 1.5.

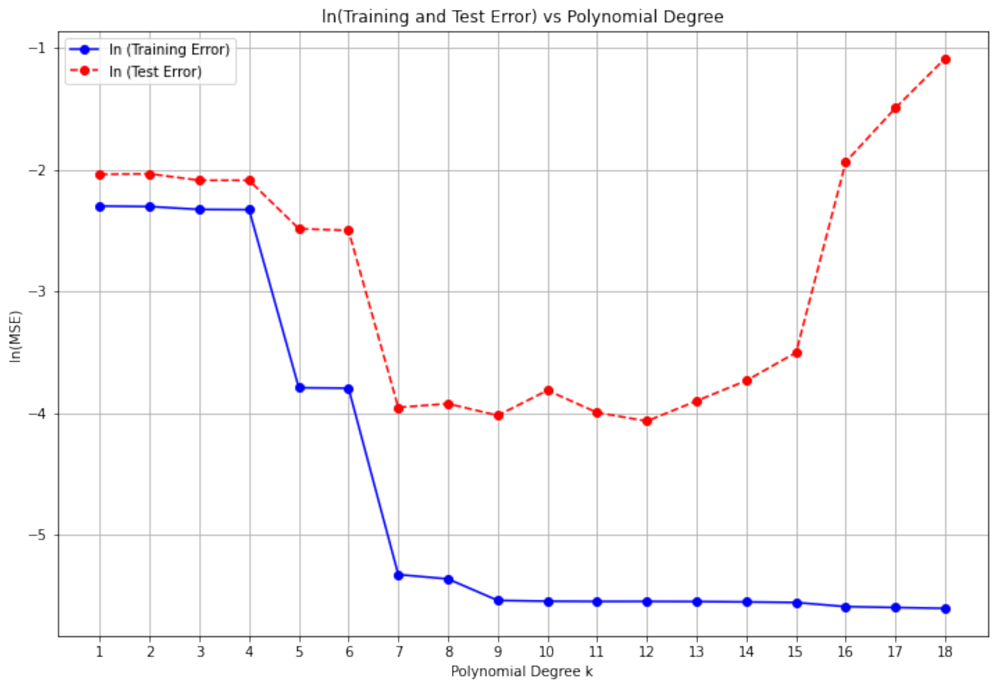

Figure 1.5. ln(Test error) versus the polynomial dimension k= 1,...,18

(d) See figure 1.6.

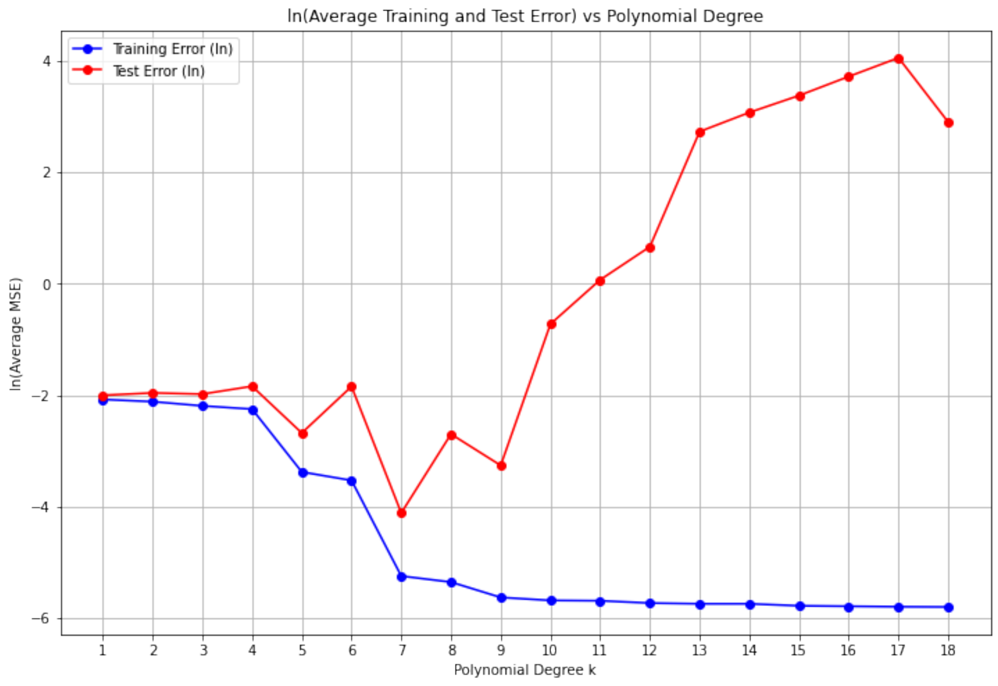

Figure 1.6. ln(average training error and testing error) versus the polynomial dimension k= 1,...,18

Question 3

(a) See figure 1.7.

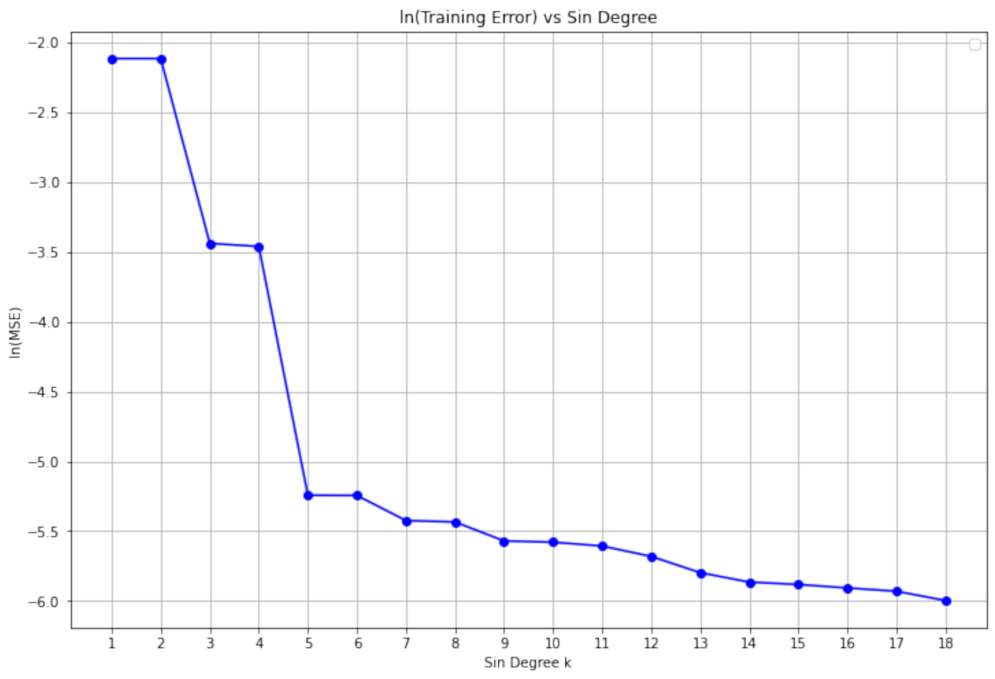

Figure 1.7. ln(training error) versus the sin function dimension k= 1,...,18

(b) See figure 1.8.

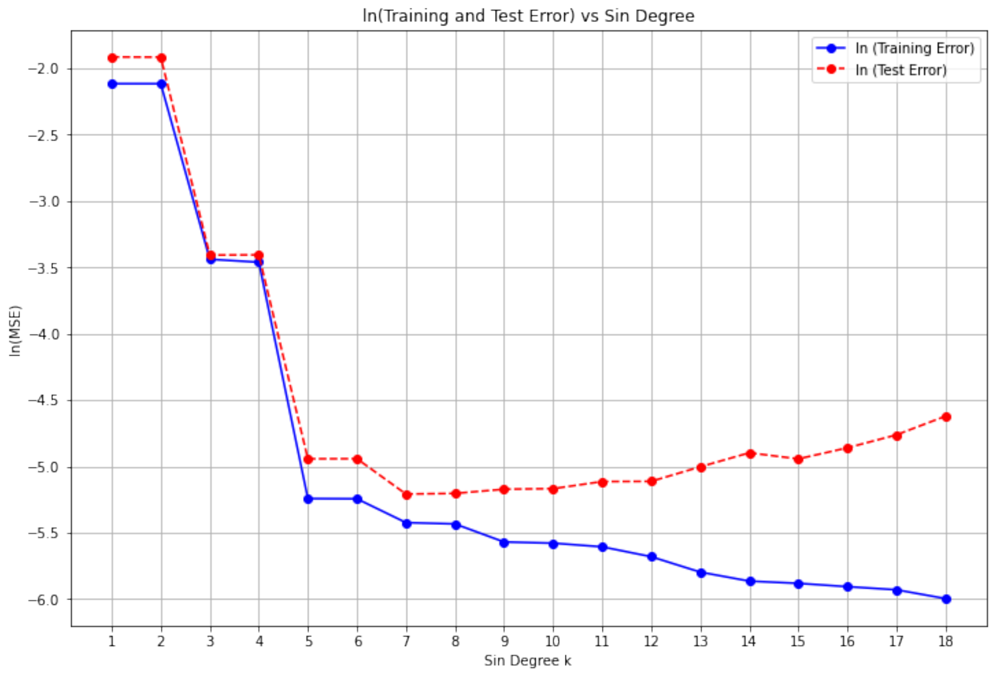

Figure 1.8. ln(Test error) versus the sin function dimension k= 1,...,18

(c) See figure 1.9.

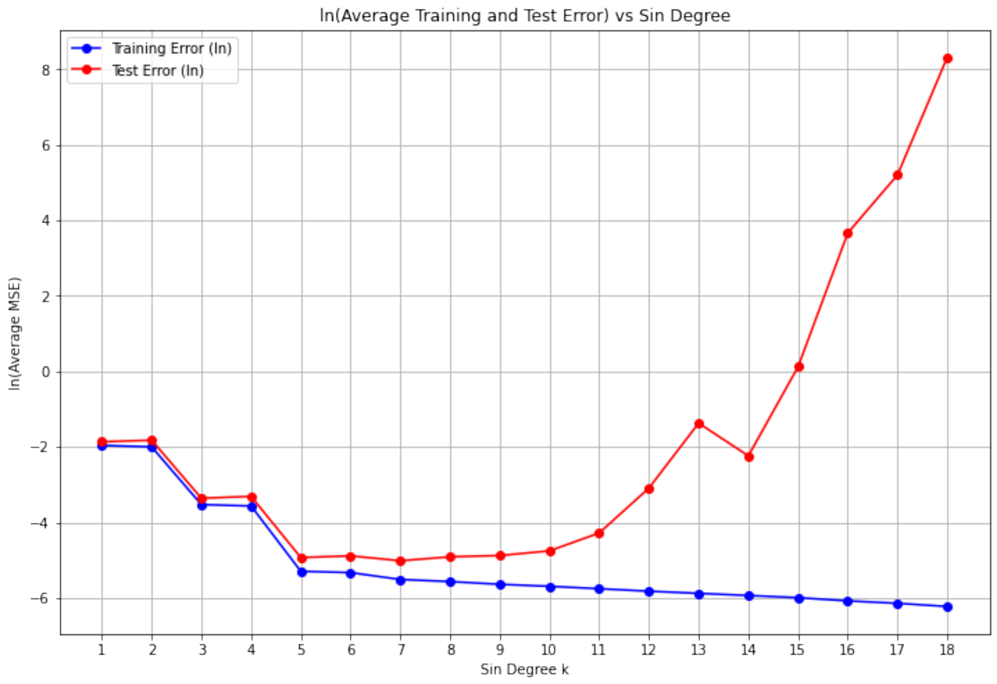

Figure 1.9. ln(average training error and testing error) versus the sin function dimension k= 1,...,18

1.2 Filtered Boston housing and kernels

Question 4

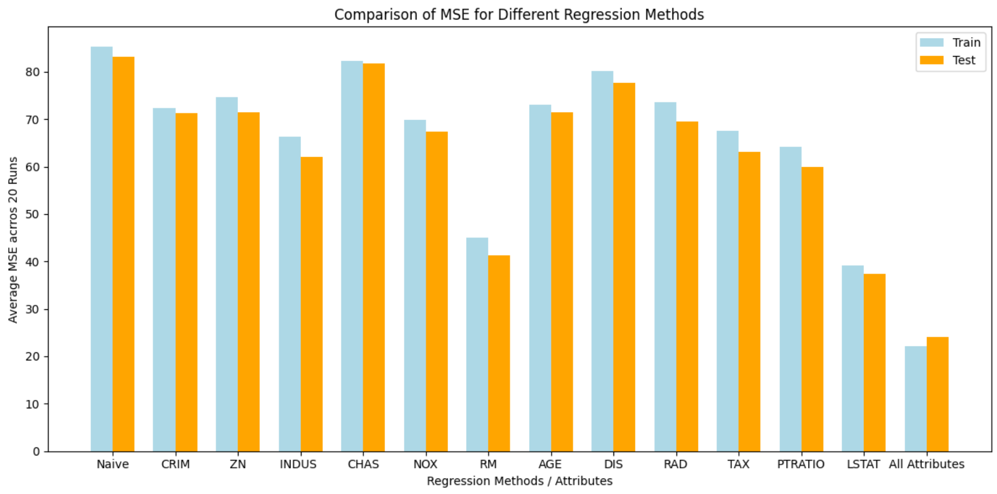

Figure 1.10. Comparison of MSE for Different Regression Methods

(a) Average MSE for Native Regression across 20 runs are as follows:

Train	Test
85.23051114281712	83.13914298510623

Table 1.1. Naive Regression Average MSE

(b) In Naive Regression, the constant function predicts the same value for all data points, regardless of the input features. The model's prediction is simply the mean of the target variable on the training set, as it provides the best possible prediction when no relationship between features and the target is modeled. As a baseline model, it can be used to measure how much improvement other models can provide. If a more complex model doesn't outperform this naive constant function, it may indicate problems with the model or the data.

(c) Average MSE for Single Attribute Regression across 20 runs are as follows:

Attribute	**Train**	**Test**
CRIM	72.38905065	71.21684028
ZN	74.63576295	71.4930145
INDUS	66.23338313	61.98658984
CHAS	82.24635271	81.82904171
NOX	69.93946866	67.4248426
RM	44.93356685	41.27891859
AGE	73.11706123	71.4877234
DIS	80.12763442	77.58419918
RAD	73.60821904	69.53502418
TAX	67.50500576	63.06659095
PTRATIO	64.22305116	59.91199375
LSTAT	39.23959731	37.38140599

Table 1.3. Single Attributes Regression Average MSE

(d) Average MSE for All Attribute Regression across 20 runs are as follows:

Train	Test
22.197098927438127	24.144534799346584

Table 1.4. All Attributes Regression Average MSE

1.3 Kernelised ridge regression

Question 5

(a) The best parameters are shown in table 1.5.

(b) See figure 1.11.

γ	σ
2^{-34}	2^{11}

Table 1.5. The Best Parameters

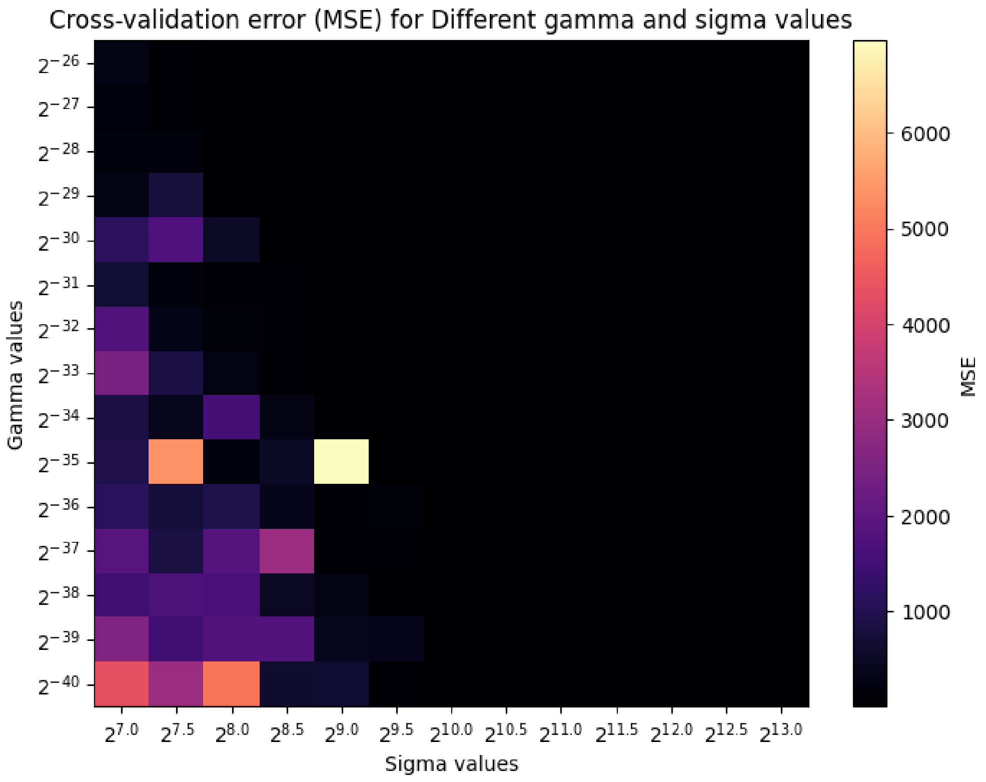

Figure 1.11. Cross-validation error as a function of γ and σ

(c) The MSE on the training and test sets for the best parameters are shown in table 1.6.

Train	Test
6.414303129538342	11.719938745567841

Table 1.6. MSE on the training and test sets using the best γ and σ

(d) See table 1.7.

Method	**MSE train**	**MSE test**
Naive Regression	84.69 ± 5.04	84.15 ± 10.05
Single Attribute Regression (attribute 1)	71.96 ± 3.97	72.85 ± 9.05
Single Attribute Regression (attribute 2)	73.67 ± 4.43	73.36 ± 8.94
Single Attribute Regression (attribute 3)	65.77 ± 4.54	62.72 ± 9.10
Single Attribute Regression (attribute 4)	82.35 ± 3.71	81.11 ± 7.44
Single Attribute Regression (attribute 5)	69.60 ± 4.71	68.11 ± 9.46
Single Attribute Regression (attribute 6)	43.75 ± 4.21	43.61 ± 8.61
Single Attribute Regression (attribute 7)	73.38 ± 4.79	70.86 ± 9.58
Single Attribute Regression (attribute 8)	79.92 ± 4.75	77.96 ± 9.66
Single Attribute Regression (attribute 9)	72.61 ± 4.44	71.52 ± 8.96
Single Attribute Regression (attribute 10)	66.84 ± 4.38	64.27 ± 8.83
Single Attribute Regression (attribute 11)	62.63 ± 4.00	63.14 ± 8.15
Single Attribute Regression (attribute 12)	38.78 ± 2.26	38.07 ± 4.51
Linear Regression (all attributes)	22.92 ± 1.72	22.74 ± 3.52
Kernel Ridge Regression	8.06 ±1.72	12.10 ± 1.99

Table 1.7. Cross-validation error as a function of γ and σ

2.1 K-Nearest Neighbors

Question 6

To generate the visualization of the hypothesis $h_{S,v}$ as shown below, I sampled 100 points x_i uniformly from $[0,1]^2$, and corresponding labels y_i were assigned uniformly at random from $\{0,1\}$. Using these labeled points as centers, I defined $h_{S,v}$ with $v = 3$ using a k-Nearest Neighbors (k-NN) algorithm where $k = v = 3$.

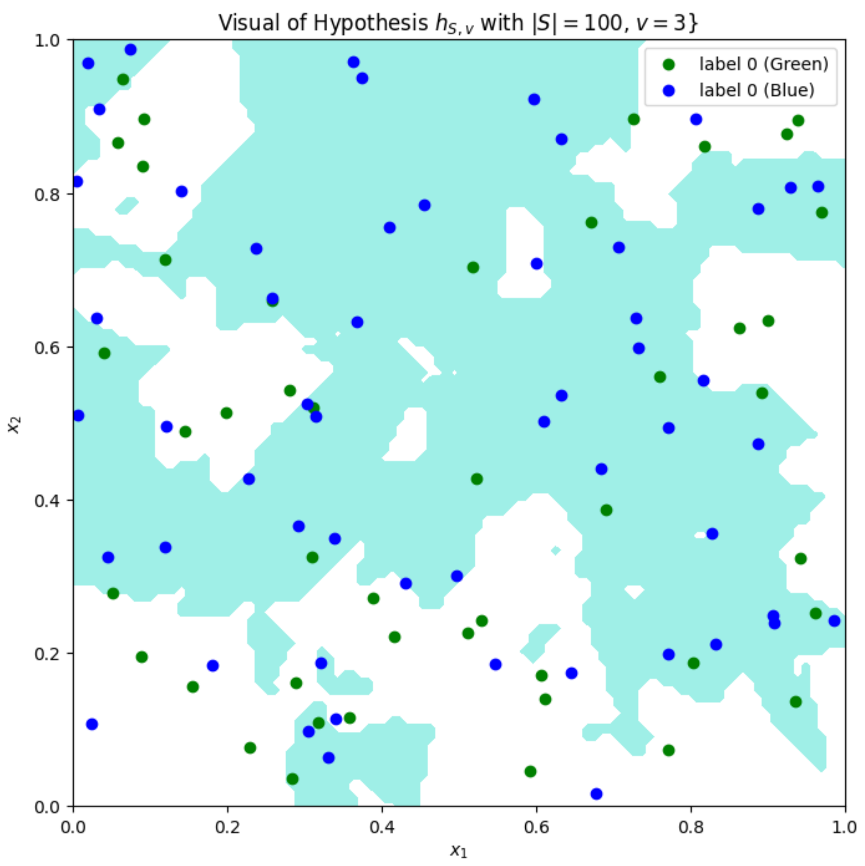

Figure 1.12. Visualization of hypothesis $h_{S,v}$ with $|S| = 100$ and $v = 3$. The white areas correspond to label 0 and the turquoise areas correspond to label 1.

For any new query point x, its label is determined by the majority label of its 3 nearest neighbors. If there is a tie in the votes, the label is set to 0. The white regions in the plot correspond to the decision boundary for $h_{S,v}$, where x is mapped to 0. Conversely, the turquoise regions represent areas where x is mapped to 1.

The labeled data points in the figure are plotted in green for label 0 and blue for label 1. This visualization demonstrates the decision boundaries and the behavior of the k-NN classifier with a small training set size ($|S| = 100$) and $v = 3$.

Question 7

We used Protocol A to compute the error for $k = 1, 2, \dots, 49$ and visualize generalization error of k-NN as a function of k. The figure below shows the estimated generalization error on the vertical axis and the value of k on the horizontal axis:

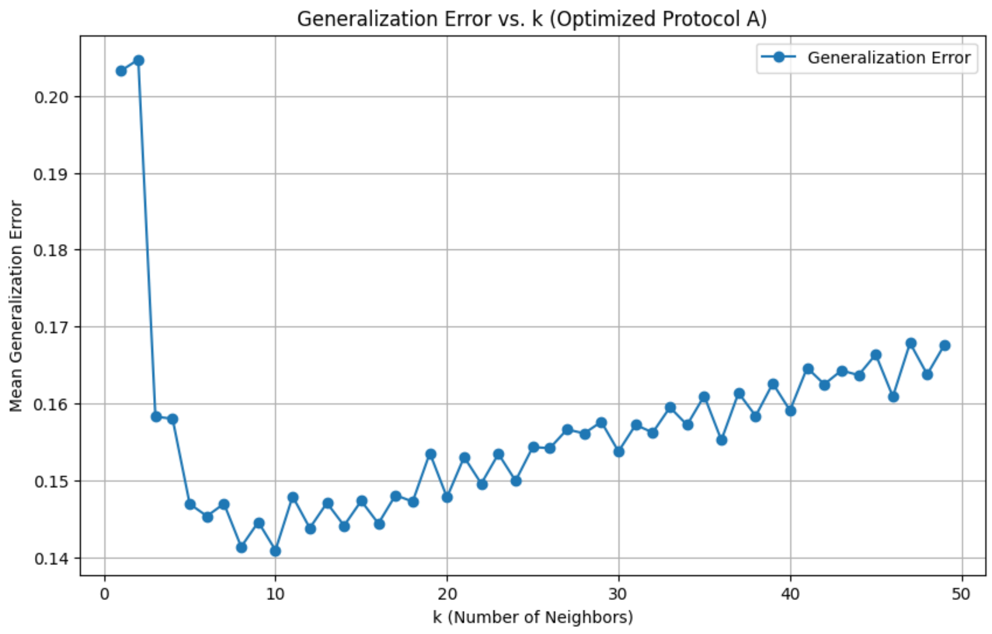

Figure 1.13. Generalization error as a function of k (Protocol A).

The generalization error demonstrates the following behavior:

- **Small k ($k \leq 5$):**
  The classifier overfits to the training data since it relies heavily on the nearest neighbors, making it highly sensitive to noise. This results in higher generalization error at very low values of k.

- **Moderate k ($5 < k \leq 10$):**
  The error decreases significantly as k increases. This is because the majority vote becomes more stable, reducing the impact of noisy or outlier points in the training data.

- **Optimal k ($10 < k \leq 20$):**
  The generalization error reaches its minimum in this range. The classifier achieves a balance between stability and locality, capturing meaningful patterns in the data.

- **Large k ($k > 20$):**
  Beyond this point, the error begins to increase gradually. Larger values of k lead to oversmoothing, where the classifier incorporates dissimilar neighbors, reducing its ability to capture local patterns.

Question 8

We used Protocol B's logic to determine the optimal k as a function of the number of training points m. The resulting plot is shown below. The x-axis represents the training set size (m), and the y-axis shows the mean optimal k across 100 runs for each m.

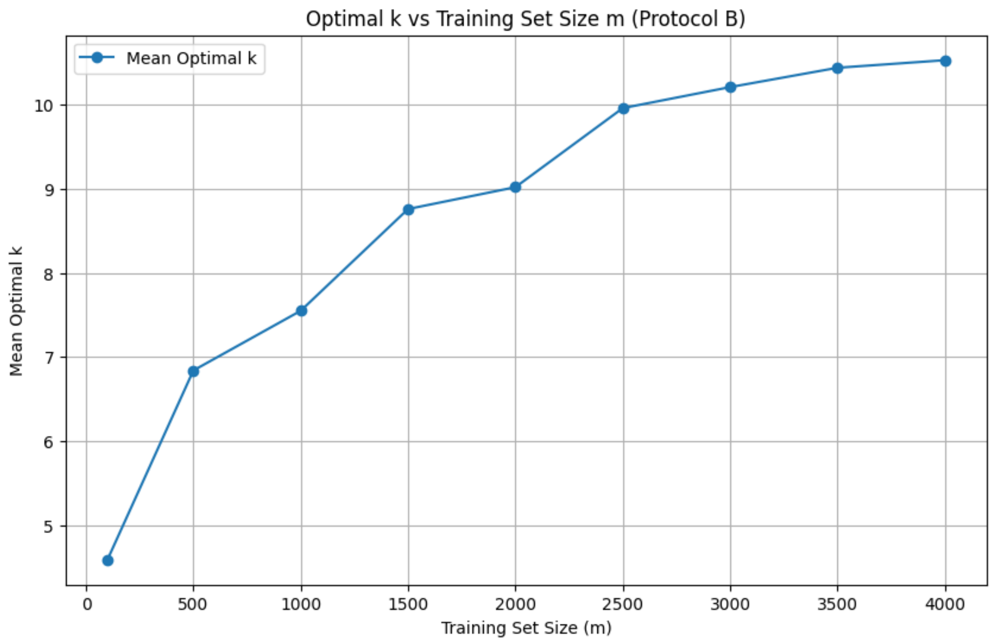

Figure 1.14. Optimal k as a function of the training set size m (Protocol B).

- **Small Training Set Sizes ($m \leq 1000$):**
  For small values of m, the optimal k is relatively low (below 5). This is because smaller training sets have fewer examples to provide reliable majority votes, making smaller neighborhoods more effective at capturing local patterns without over-smoothing.
- **Moderate Training Set Sizes ($1000 < m \leq 2500$):**
  As the training set size increases, the mean optimal k grows steadily. This indicates that larger training sets provide more data for stable majority voting, allowing the classifier to expand the neighborhood size (k) without introducing significant noise.
- **Large Training Set Sizes ($m > 2500$):**
  For larger training set sizes, the mean optimal k increases more slowly and appears to plateau at around $k = 10$. At this point, the classifier balances incorporating a sufficient number of neighbors for stability while avoiding excessive smoothing that might obscure local patterns.

3.1 Questions

Question 9

(a) A kernel $K(x, z)$ is positive semidefinite (PSD) if, for any finite set of points $\{x_1, \ldots, x_n\}$, the Gram matrix $\mathbf{K}$ defined by $\mathbf{K}_{ij} = K(x_i, x_j)$ satisfies $\mathbf{v}^T\mathbf{K}\mathbf{v} \geq 0$ for any non-zero vector $\mathbf{v}$.

For $K_c(x, z) = c + \sum_{i=1}^{n} x_i z_i$, the Gram matrix is:

$$\mathbf{K} = \mathbf{K}^{\text{linear}} + c\mathbf{1},$$

where $\mathbf{K}_{ij}^{\text{linear}} = \langle x_i, x_j \rangle$ is the Gram matrix of the linear kernel and $\mathbf{1}$ is the identity matrix.

1: Linear kernel is PSD. The Gram matrix $\mathbf{K}^{\text{linear}}$ is PSD because the linear kernel represents an inner product.

2: Adding $c\mathbf{1}$. Adding $c\mathbf{1}$ increases one eigenvalue of $\mathbf{K}$ by cn (where n is the number of data points) and does not affect the non-zero eigenvalues of $\mathbf{K}^{\text{linear}}$.

3: Ensure PSD. For $\mathbf{K}$ to remain PSD, the smallest eigenvalue must be non-negative. Since $\mathbf{K}^{\text{linear}}$ is PSD, this condition is satisfied if $c \geq 0$.

Hence, $K_c(x, z)$ is PSD if $c \geq 0$.

(b) In kernelized linear regression, the solution is given by:

$$\mathbf{w} = (\mathbf{K} + \lambda \mathbf{I})^{-1}\mathbf{y},$$

where $\mathbf{K}$ is the Gram matrix using K_c.

Effect of c:

- Increasing c adds a uniform value to all entries of $\mathbf{K}$, increasing its diagonal dominance and making the solution smoother and less sensitive to individual data points.
- Smaller c (with $c \geq 0$) emphasizes the linear component $\langle x, z \rangle$, leading to a more flexible model that captures finer details of the data.

We can conclude that c controls the trade-off between regularization and sensitivity to data. Larger c smoothens the solution, while smaller c makes it more flexible.

Question 10

To simulate a 1-Nearest Neighbor (1-NN) classifier, the parameter β in the Gaussian kernel must be chosen large enough. As $\beta \to \infty$, the kernel $K_\beta(x, t)$ becomes highly localized, focusing on the closest training point x_i to the test point t. In this case, the linear classifier behaves similarly to a 1-NN classifier because the decision is based on the label of the nearest neighbor.

We are given the kernel function:

$$K_\beta(x, t) = \exp(-\beta \|x - t\|^2)$$

and the classifier function:

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

We want to show that as $\beta \to \infty$, the classifier behaves like a 1-Nearest Neighbor (1-NN) classifier.

We first look at the behavior of the Gaussian Kernel as $\beta \to \infty$:

As $\beta \to \infty$, the Gaussian kernel $K_\beta(x, t) = \exp(-\beta \|x - t\|^2)$ becomes highly localized around $x = t$. Specifically, for any fixed t, if $\|x - t\|^2$ is non-zero, the kernel value $K_\beta(x, t)$

tends to zero exponentially fast as β increases. The only point where the kernel is non-zero is when $x = t$, i.e.

$$\lim_{\beta\to\infty} K_\beta(x,t) = \begin{cases} 1, & \text{if } x = t \\ 0, & \text{if } x \neq t \end{cases}$$

We then analyze the behavior of the classifier function as $\beta \to \infty$:

The classifier function is given by:

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

where α_i are the weights learned during the linear regression process, representing the contributions of the training points to the prediction.

As $\beta \to \infty$, the kernel $K_\beta(x_i,t)$ is significant only for the point x_i that is closest to t. Let x_{nearest} be the closest training point to t, i.e., $x_{\text{nearest}} = \arg\min_i \|x_i - t\|$.

For x_{nearest}, we have:

$$K_\beta(x_{\text{nearest}}, t) \to 1 \quad \text{as} \quad \beta \to \infty$$

For all other points $x_i \neq x_{\text{nearest}}$, the kernel values $K_\beta(x_i,t) \to 0$.

Thus, the classifier function $f(t)$ becomes:

$$f(t) = \alpha_{\text{nearest}} K_\beta(x_{\text{nearest}}, t) + \sum_{i \neq \text{nearest}} \alpha_i K_\beta(x_i, t)$$

As $\beta \to \infty$, the second sum tends to zero, so:

$$f(t) \approx \alpha_{\text{nearest}}$$

Finally, we check the decision rule:

The classifier makes a decision based on the sign of $f(t)$, i.e.:

$$\hat{y}(t) = \text{sign}(f(t))$$

As $f(t) \approx \alpha_{\text{nearest}}$, the sign of $f(t)$ is determined by the label associated with the nearest training point x_{nearest}, making the decision rule equivalent to a 1-NN classifier.

To conclude, we found that:

As $\beta \to \infty$, the linear classifier with the Gaussian kernel $K_\beta(x,t) = \exp(-\beta\|x-t\|^2)$ approximates a 1-NN classifier because the kernel function becomes sharply localized around the closest training point, making the decision based solely on the label of the nearest neighbor.

Question 11

(a) Let $C \subset X$ be a subset of X with cardinality $|C| = 2n$. Denote by $\mathcal{Y}^C = \{f_1, \ldots, f_T\}$ the set of all possible functions from $C \to Y$. We define the distribution ρ_i for each f_i as follows:

$$\rho_i(\{(x,y)\}) = \begin{cases} \frac{1}{2n}, & \text{if } y = f_i(x) \\ 0, & \text{otherwise} \end{cases}$$

The excess risk for a classifier f is defined as:

$$E_\rho(f) = \int_{X\times Y} 1\{f(x) \neq y\}\, d\rho(x,y)$$

Since $f_i(x)$ correctly classifies all points in C, we have $f_i(x) = y$ for all $x \in C$, and the probability assigned to any misclassified pair is zero. Therefore, the classifier f_i minimizes the excess risk for the distribution ρ_i.

Thus, we conclude that $E_{\rho_i}(f_i) = \inf_{f:X\to Y} E_\rho(f) = 0$.

(b) Let C^n be the set of all possible input datasets of size n, with $k = |C^n| = (2n)^n$. For every $j = 1,\dots,k$ and input set $S_j = (x_1,\dots,x_n) \in C^n$, denote $S_j^i = \{(x_1, f_i(x_1)),\dots,(x_n, f_i(x_n))\}$ the corresponding input-output dataset for each $i = 1,\dots,T$.

We aim to show that:

$$\max_{i=1,\dots,T} \mathbb{E}_{S\sim\rho_i}\mathbb{E}_{\rho_i}(A(S)) \geq \min_{j=1,\dots,k} \frac{1}{T}\sum_{i=1}^{T} \mathbb{E}_{\rho_i}(A(S_j^i))$$

We first apply the hint to the mis-classification risks:

Using the hint that for any set of scalars $\alpha_1,\dots,\alpha_m$, we have:

$$\max_\ell \alpha_\ell \geq \frac{1}{m}\sum_{\ell=1}^{m} \alpha_\ell \geq \min_\ell \alpha_\ell$$

we apply this to the misclassification risks for the different input datasets. Specifically, we define $\alpha_i = \mathbb{E}_{S\sim\rho_i}\mathbb{E}_{\rho_i}(A(S))$ for each i, and we have:

$$\max_{i=1,\dots,T} \mathbb{E}_{S\sim\rho_i}\mathbb{E}_{\rho_i}(A(S)) \geq \frac{1}{T}\sum_{i=1}^{T} \mathbb{E}_{S\sim\rho_i}\mathbb{E}_{\rho_i}(A(S)) \geq \min_{i=1,\dots,T} \mathbb{E}_{S\sim\rho_i}\mathbb{E}_{\rho_i}(A(S))$$

Then, we relate the hint to the average risk over datasets:

Now, applying the second part of the hint, we see that:

$$\frac{1}{T}\sum_{i=1}^{T} \mathbb{E}_{S\sim\rho_i}\mathbb{E}_{\rho_i}(A(S)) \geq \min_{j=1,\dots,k} \frac{1}{T}\sum_{i=1}^{T} \mathbb{E}_{\rho_i}(A(S_j^i))$$

Thus, we have shown the desired inequality:

$$\max_{i=1,\dots,T} \mathbb{E}_{S\sim\rho_i}\mathbb{E}_{\rho_i}(A(S)) \geq \min_{j=1,\dots,k} \frac{1}{T}\sum_{i=1}^{T} \mathbb{E}_{\rho_i}(A(S_j^i))$$

(c) Let $S_j = \{(x_1, \ldots, x_n)\}$ and $R_j = \{v_1, \ldots, v_p\}$ be the subset of points of C that do not belong to S_j. We need to show the inequality:

$$\frac{1}{T}\sum_{i=1}^{T} \mathbb{E}_{\rho_i}(A(S_j^i)) \geq \frac{1}{2}\min_{v\in R_j} \frac{1}{T}\sum_{i=1}^{T} 1\{A(S_j^i)(v) \neq f_i(v)\}$$

We can lower bound the risk by considering only the errors over the set R_j (the points not in S_j):

$$\frac{1}{T}\sum_{i=1}^{T} \mathbb{E}_{\rho_i}(A(S_j^i)) \geq \frac{1}{T}\sum_{i=1}^{T} 1\{A(S_j^i)(v) \neq f_i(v)\}, \quad \forall v \in R_j$$

By the hint, we use the fact that the average error over all points in R_j is at least the minimum error over a specific point $v \in R_j$, i.e.:

$$\frac{1}{T}\sum_{i=1}^{T} 1\{A(S_j^i)(v) \neq f_i(v)\} \geq \min_{v\in R_j} \frac{1}{T}\sum_{i=1}^{T} 1\{A(S_j^i)(v) \neq f_i(v)\}$$

Combining these two results, we obtain the desired inequality:

$$\frac{1}{T}\sum_{i=1}^{T} \mathbb{E}_{\rho_i}(A(S_j^i)) \geq \frac{1}{2}\min_{v\in R_j} \frac{1}{T}\sum_{i=1}^{T} 1\{A(S_j^i)(v) \neq f_i(v)\}$$

(d) Let $v \in R_j$ be a point in the subset R_j, and let S_j be the corresponding set of input-output pairs. We aim to show that:

$$\frac{1}{T}\sum_{i=1}^{T} 1\{A(S_j^i)(v) \neq f_i(v)\} = \frac{1}{2}$$

By the hint, we can partition $\mathcal{Y}^C$ into $T/2$ pairs $(f_i, f_{i'})$, where for each pair $(f_i, f_{i'})$, the functions f_i and $f_{i'}$ differ at exactly one point: $x = v$. In other words, $f_i(x) \neq f_{i'}(x)$ if and only if $x = v$.

For each pair $(f_i, f_{i'})$, the classifiers will misclassify v with probability $1/2$, because half the time $A(S_j^i)(v)$ will equal $f_i(v)$ and the other half it will not.

Thus, for each pair $(f_i, f_{i'})$, the misclassification indicator $1\{A(S_j^i)(v) \neq f_i(v)\}$ will be 1 for half of the cases and 0 for the other half.

Since we have $T/2$ pairs of functions, and for each pair the misclassification rate is $1/2$, the average misclassification rate across all T classifiers is:

$$\frac{1}{T}\sum_{i=1}^{T} 1\{A(S_j^i)(v) \neq f_i(v)\} = \frac{1}{2}$$

Therefore, we have shown that for any $v \in R_j$, the misclassification rate at v across all classifiers is $\frac{1}{2}$.

(e) Let Z be a random variable with values in the interval $[0, 1]$, and let $E[Z] = \mu$. We are tasked with proving the following inequality for any $a \in (0, 1)$:

$$\mathbb{P}(Z > 1 - a) \geq \frac{\mu - (1 - a)}{a}$$

We will use **Markov's inequality**, which states that for any non-negative random variable X and any $t > 0$, we have:

$$\mathbb{P}(X \geq t) \leq \frac{E[X]}{t}$$

Let $X = Z$ and $t = 1 - a$. By Markov's inequality, we get:

$$\mathbb{P}(Z \geq 1 - a) \leq \frac{E[Z]}{1 - a} = \frac{\mu}{1 - a}$$

We are interested in $\mathbb{P}(Z > 1 - a)$. Since Z is a continuous random variable, we have $\mathbb{P}(Z = 1 - a) \approx 0$, so:

$$\mathbb{P}(Z > 1 - a) \geq \mathbb{P}(Z \geq 1 - a)$$

Thus, from Markov's inequality, we get:

$$\mathbb{P}(Z > 1 - a) \geq \frac{\mu}{1 - a}$$

We now rewrite the right-hand side as:

$$\frac{\mu}{1 - a} = \frac{\mu - (1 - a)}{a}$$

Thus, we have shown that:

$$\mathbb{P}(Z > 1 - a) \geq \frac{\mu - (1 - a)}{a}$$

as required.

(f) We are tasked with proving that for any algorithm A and any integer n, there exists a distribution ρ over $X \times Y$ such that:

$$P_{S \sim \rho^n}\left(\mathbb{E}_\rho(A(S)) > \frac{1}{8}\right) \geq \frac{1}{7}$$

We will combine the results from previous exercises to establish this bound.

First, we apply the markov's inequality:

We start by applying Markov's inequality to the random variable $Z = \mathbb{E}_\rho(A(S))$. By Markov's inequality, for any $t > 0$, we have:

$$P(Z \geq t) \leq \frac{E[Z]}{t}$$

In our case, $Z = \mathbb{E}_\rho(A(S))$, and we wish to bound the probability $P(Z > \frac{1}{8})$.

We then use the previous bound for Z:

From the results in part (e), we know that for $Z = \mathbb{E}_\rho(A(S))$, we can lower bound the probability $P(Z > 1 - a)$ using the expected value $\mu = E[Z]$. Specifically, we showed that:

$$P(Z > 1 - a) \geq \frac{\mu - (1 - a)}{a}$$

We now choose $a = \frac{7}{8}$ to obtain the desired bound:

$$P(Z > \frac{1}{8}) \geq \frac{\mu - \frac{7}{8}}{\frac{7}{8}} = \frac{\mu - \frac{7}{8}}{\frac{7}{8}}$$

Finally, combine the Results

Using the previous results, we can conclude that for $a = \frac{7}{8}$, we have:

$$P_{S \sim \rho^n}\left(\mathbb{E}_\rho(A(S)) > \frac{1}{8}\right) \geq \frac{1}{7}$$

Thus, we have shown the desired inequality.

(g) **(i) Formal Statement of the Theorem (No-Free-Lunch):**

The No-Free-Lunch (NFL) theorem states that for any algorithm A and any distribution ρ over $X \times Y$, there exists a distribution ρ such that for all $\epsilon \in (0, 1)$, $\delta \in (0, 1)$, and any integer $n \geq n_0$, there exists a dataset S where:

$$P_{S \sim \rho^n}\left(\mathbb{E}_\rho(A(S)) - \inf_{f \in \mathcal{H}} \mathbb{E}_\rho(f) \leq \epsilon\right) \geq 1 - \delta$$

This implies that no single algorithm can achieve universally good performance across all possible distributions.

(ii) Comparison with the Definition of Learnable Space:

As described, the definition of a learnable space of functions $\mathcal{H}$ requires that there exists an algorithm A that can achieve good generalization (error $\leq \epsilon$) for all distributions ρ, given sufficient sample size $n \geq \bar{n}$.

To compare, the key difference lies in the scope of the function space being considered:

- **NFL Theorem:** This applies to the space of all possible functions, $\mathcal{Y}^{\mathcal{X}}$. This space is too broad, including functions that may be arbitrarily complex, random, or unrelated to the given task. Hence, $\mathcal{Y}^{\mathcal{X}}$ is not learnable, as no algorithm can generalize well over all possible functions in this space.
- **Learnable Space:** This focuses on specific subsets of functions ($\mathcal{H}$) that are typically structured, well-behaved, or task-specific. For a function space $\mathcal{H}$ to be learnable, it must allow for an algorithm to generalize well across all distributions ρ with a sufficiently large dataset.

The NFL theorem does imply that $\mathcal{Y}^{\mathcal{X}}$ is not learnable because the space is too large and diverse to satisfy the learnability criteria. However, the learnability definition applies to smaller, constrained function spaces $\mathcal{H}$ that are often chosen to match specific problem domains.

(iii) Implications for Machine Learning Algorithm Design:

The No-Free-Lunch theorem implies the following:

- There is no universal best algorithm; the choice of algorithm depends on the task and dataset.
- Algorithms must be tailored to the specific characteristics of the problem at hand.
- Model selection requires careful validation and cannot rely on theoretical guarantees alone.